A Sequential Matching Framework for Multi-Turn Response Selection in Retrieval-Based Chatbots

Yu Wu
Beihang University
State Key Laboratory of Software Development Environment
wuyu@buaa.edu.cn

Wei Wu
Microsoft Corporation
Research and AI Group
wuwei@microsoft.com

Chen Xing
NanKai University
College of Computer and Control Engineering
xingchen1113@gmail.com

Can Xu
Microsoft Corporation
Research and AI Group
can.xu@microsoft.com

Zhoujun Li
Beihang University
State Key Laboratory of Software Development Environment
lizj@buaa.edu.cn

Ming Zhou
Microsoft Research
Natural Language Computing Group
mingzhou@microsoft.com

Submission received: 14 August 2017; revised version received: 3 July 2018; accepted for publication: 6 December 2018.

doi:10.1162/COLI_a_00345

We study the problem of response selection for multi-turn conversation in retrieval-based chatbots. The task involves matching a response candidate with a conversation context, the challenges for which include how to recognize important parts of the context, and how to model the relationships among utterances in the context. Existing matching methods may lose important information in contexts as we can interpret them with a unified framework in which contexts are transformed to fixed-length vectors without any interaction with responses before matching. This motivates us to propose a new matching framework that can sufficiently carry important information in contexts to matching and model relationships among utterances at the same time. The new framework, which we call a sequential matching framework (SMF), lets each utterance in a context interact with a response candidate at the first step and transforms the pair to a matching vector. The matching vectors are then accumulated following the order of the utterances in the context with a recurrent neural network (RNN) that models relationships among utterances. Context-response matching is then calculated with the hidden states of the RNN. Under SMF, we propose a sequential convolutional network and sequential attention network and conduct experiments on two public data sets to test their performance. Experiment results show that both models can significantly outperform state-of-the-art matching methods. We also show that the models are interpretable with visualizations that provide us insights on how they capture and leverage important information in contexts for matching.

1. Introduction

Recent years have witnessed a surge of interest on building conversational agents both in industry and academia. Existing conversational agents can be categorized into task-oriented dialog systems and non–task-oriented chatbots. Dialog systems focus on helping people complete specific tasks in vertical domains (Young et al. 2010), such as flight booking, bus route enquiry, restaurant recommendation, and so forth; chatbots aim to naturally and meaningfully converse with humans on open domain topics (Ritter, Cherry, and Dolan 2011). Building an open domain chatbot is challenging, because it requires the conversational engine to be capable of responding to any input from humans that covers a wide range of topics. To address the problem, researchers have considered leveraging the large amount of conversation data available on the Internet, and proposed generation-based methods (Shang, Lu, and Li 2015; Vinyals and Le 2015; Li et al. 2016b; Mou et al. 2016; Serban et al. 2016; Xing et al. 2017) and retrieval-based methods (Wang et al. 2013; Hu et al. 2014; Ji, Lu, and Li 2014; Wang et al. 2015; Yan, Song, and Wu 2016; Zhou et al. 2016; Wu et al. 2018a). Generation-based methods generate responses with natural language generation models learned from conversation data, while retrieval-based methods re-use the existing responses by selecting proper ones from an index of the conversation data. In this work, we study the problem of response selection in retrieval-based chatbots, because retrieval-based chatbots have the advantage of returning informative and fluent responses. Although most existing work on retrieval-based chatbots studies response selection for single-turn conversation (Wang et al. 2013) in which conversation history is ignored, we study the problem in a multi-turn scenario. In a chatbot, multi-turn response selection takes a message and utterances in its previous turns as an input and selects a response that is natural and relevant to the entire context.

A key step in response selection is measuring matching degree between an input and response candidates. Different from single-turn conversation, in which the input is a single utterance (i.e., the message), multi-turn conversation requires context-response

matching where both the current message and the utterances in its previous turns should be taken into consideration. The challenges of the task include (1) how to extract important information (words, phrases, and sentences) from the context and leverage the information in matching; and (2) how to model relationships and dependencies among the utterances in the context. Table 1 uses an example to illustrate the challenges. First, to find a proper response for the context, the chatbot must know that "hold a drum class" and "drum" are important points. Without them, it may return a response relevant to the message (i.e., Turn-5 in the context) but nonsensical in the context (e.g., "what lessons do you want?"). On the other hand, words like "Shanghai" and "Lujiazui" are less useful and even noisy to response selection. The responses from the chatbot may drift to the topic of "Shanghai" if the chatbot pays significant attention to these words. Therefore, it is crucial yet non-trivial to let the chatbot understand the important points in the context and leverage them in matching and at the same time circumvent noise. Second, there is a clear dependency between Turn-5 and Turn-2 in the context, and the order of utterances matters in response selection because there will be different proper responses if we exchange Turn-3 and Turn-5.

Existing work, including the recurrent neural network architectures proposed by Lowe et al. (2015), the deep learning to respond architecture proposed by Yan, Song, and Wu (2016), and the multi-view architecture proposed by Zhou et al. (2016), may lose important information in context-response matching because they follow the same paradigm to perform matching, which suffers clear drawbacks. In fact, although these models have different structures, they can be interpreted with a unified framework: A context and a response are first individually represented as vectors, and then their matching score is computed with the vectors. The context representation includes two layers. The first layer represents utterances in the context, and the second layer takes the output of the first layer as an input and represents the entire context. The existing work differs in how they design the context representation and the response representation and how they calculate the matching score with the two representations. The framework view unifies the existing models and indicates the common drawbacks they have: everything in the context is compressed to one or more fixed-length vectors before matching is conducted; and there is no interaction between the context and the response in the formation of their representations. The context is represented without enough supervision from the response, and so is the response.

To overcome the drawbacks, we propose a sequential matching network (SMN) for context-response matching in our early work (Wu et al. 2017) where we construct

Table 1
An example of multi-turn conversation.

	Context
Turn-1	*Human*: How are you doing?
Turn-2	*ChatBot*: I am going to **hold a drum class** in Shanghai. Anyone wants to join? The location is near Lujiazui.
Turn-3	*Human*: Interesting! Do you have coaches who can help me practice **drum**?
Turn-4	*ChatBot*: Of course.
Turn-5	*Human*: Can I have a free first lesson?
	Response Candidates
	Response 1: Sure. Have you ever played drum before? ✓
	Response 2: What lessons do you want? ✗

a matching vector for each utterance–response pair through convolution and pooling on their similarity matrices, and then aggregate the sequence of matching vectors as a matching score of the context and the response. In this work, we take it one step further and generalize the SMN model to a sequential matching framework (SMF). The framework view allows us to tackle the challenges of context-response matching from a high level. Specifically, SMF matches each utterance in the context with the response at the first step and forms a sequence of matching vectors. It then accumulates the matching vectors of utterance–response pairs in the chronological order of the utterances. The final context-response matching score is calculated with the accumulation of pair matching. Different from the existing framework, SMF allows utterances in the context and the response to interact with each other at the very beginning, and thus important matching information in each utterance–response pair can be sufficiently preserved and carried to the final matching score. Moreover, relationships and dependencies among utterances are modeled in a matching fashion, so the order of utterances can supervise the aggregation of the utterance–response matching. Generally speaking, SMF consists of three layers. The first layer extracts important matching information from each utterance–response pair and transforms the information into a matching vector. The matching vectors are then uploaded to the second layer where a recurrent neural network with gated recurrent units (GRU) (Chung et al. 2014) is used to model the relationships and dependencies among utterances and accumulate the matching vectors into its hidden states. The final layer takes the hidden states of the GRU as input and calculates a matching score for the context and the response.

The key to the success of SMF lies in how to design the utterance–response matching layer, which requires identification of important parts in each utterance. We first show that the point-wise similarity calculation followed by convolution and pooling in SMN is one implementation of the utterance–response matching layer of SMF, making the SMN model a special case of the framework. Then, we propose a new model named sequential attention network (SAN), which implements the utterance–response matching layer of SMF with an attention mechanism. Specifically, for an utterance–response pair, SAN lets the response attend to important parts (either words or segments) in the utterance by weighting the parts using each part of the response. Each weight reflects how important the part in the utterance is with respect to the corresponding part in the response. Then for each part in the response, parts in the utterance are linearly combined with the weights, and the combination interacts with the part of the response by Hadamard product to form a representation of the utterance. Such utterance representations are computed on both a word level and a segment level. The two levels of representations are finally concatenated and processed by a GRU to form a matching vector. SMN and SAN are two different implementations of the utterance–response matching layer, and we give a comprehensive comparison between SAN and SMN. Theoretically, SMN is faster and easier to parallelize than SAN, whereas SAN can better utilize the sequential relationship and dependency. The empirical results are consistent with the theoretical analysis.

We empirically compare SMN and SAN on two public data sets: the Ubuntu Dialogue Corpus (Lowe et al. 2015) and the Douban Conversation Corpus (Wu et al. 2017). The Ubuntu corpus is a large-scale English data set in which negative instances are randomly sampled and dialogues are collected from a specific domain; the Douban corpus is a newly published Chinese data set where conversations are crawled from an open domain forum with response candidates collected following the procedure of retrieval-based chatbots and their appropriateness judged by human annotators. Experimental results show that on both data sets, both SMN and SAN can significantly

outperform the existing methods. Particularly, on the Ubuntu corpus, SMN and SAN yield 6 and 7 percentage point improvement, respectively, on R_{10}@1 over the best performing baseline method, and on the Douban corpus, the improvement on mean average precision from SMN and SAN over the best baseline are 2.6 and 3.6 percentage points, respectively. The empirical results indicate that SAN can achieve better performance than SMN in practice. In addition to the quantitative evaluation, we also visualize the two models with examples from the Ubuntu corpus. The visualization reveals how the two models understand conversation contexts and provides us insights on why they can achieve big improvement over state-of-the-art methods.

This work is a substantial extension of our previous work reported at ACL 2017. The extension in this article includes a unified framework for the existing methods, a proposal of a new framework for context-response matching, and a new model under the framework. Specifically, the contributions of this work include the following.

- We unify existing context-response matching models with a framework and disclose their intercorrelations with detailed mathematical derivations, which reveals their common drawbacks and sheds light on our new direction.
- We propose a new framework for multi-turn response selection, namely, the sequential matching framework, which is capable of overcoming the drawbacks suffered by the existing models and addressing the challenges of context-response matching in an end-to-end way. The framework indicates that the key to context-response matching is not the 2D convolution and pooling operations in SMN, but a general utterance–response matching function that can capture the important matching information in utterance–response pairs.
- We propose a new architecture, the sequential attention network, under the new framework. Moreover, we compare SAN with SMN on both efficiency and effectiveness.
- We conduct extensive experiments on public data sets and verify that SAN achieves new state-of-the-art performance on context-response matching.

The rest of the paper is organized as follows: In Section 2 we summarize the related work. We formalize the learning problem in Section 3. In Section 4, we interpret the existing models with a framework. Section 5 elaborates our new framework and gives two models as special cases of the framework. Section 6 gives the learning objective and some training details. In Section 7 we give details of the experiments. In Section 8, we outline our conclusions.

2. Related Work

We briefly review the history and recent progress of chatbots, and application of text matching techniques in other tasks. Together with the review of existing work, we clarify the connection and difference between these works and our work in this article.

2.1 Chatbots

Research on chatbots goes back to the 1960s when ELIZA (Weizenbaum 1966), an early chatbot, was designed with a large number of handcrafted templates and heuristic rules.

ELIZA needs huge human effort but can only return limited responses. To remedy this, researchers have developed data-driven approaches (Higashinaka et al. 2014). The idea behind data-driven approaches is to build a chatbot with the large amount of conversation data available on social media such as forums and microblogging services. Methods along this line can be categorized into retrieval-based and generation-based ones.

Generation-based chatbots reply to a message with natural language generation techniques. Early work (Ritter, Cherry, and Dolan 2011) regards messages and responses as source language and target language, respectively, and learn a phrase-based statistical machine translation model to translate a message to a response. Recently, together with the success of deep learning approaches, the sequence-to-sequence framework has become the mainstream approach, because it can implicitly capture compositionality and long-span dependencies in languages. Under this framework, many models have been proposed for both single-turn conversation and multi-turn conversation. For example, in single-turn conversation, sequence-to-sequence with an attention mechanism (Shang, Lu, and Li 2015; Vinyals and Le 2015) has been applied to response generation; Li et al. (2016a) proposed a maximum mutual information objective to improve diversity of generated responses; Xing et al. (2017) and Mou et al. (2016) introduced external knowledge into the sequence-to-sequence model; Wu et al. (2018b) proposed decoding a response from a dynamic vocabulary; Li et al. (2016b) incorporated persona information into the sequence-to-sequence model to enhance response consistency with speakers; and Zhou et al. (2018) explored how to generate emotional responses with a memory augmented sequence-to-sequence model. In multi-turn conversation, Sordoni et al. (2015) compressed a context to a vector with a multi-layer perceptron in response generation; Serban et al. (2016) extended the sequence-to-sequence model to a hierarchical encoder-decoder structure; and under this structure, they further proposed two variants including VHRED (Serban et al. 2017b) and MrRNN (Serban et al. 2017a) to introduce latent and explicit variables into the generation process. Xing et al. (2018) exploited a hierarchical attention mechanism to highlight the effect of important words and utterances in generation. Upon these methods, reinforcement learning technique (Li et al. 2016c) and an adversarial learning technique (Li et al. 2017) have also been applied to response generation.

Different from the generation based systems, retrieval-based chatbots select a proper response from an index and re-use the one to reply to a new input. The key to response selection is how to match the input with a response. In a single-turn scenario, matching is conducted between a message and a response. For example, Hu et al. (2014) proposed message-response matching with convolutional neural networks; Wang et al. (2015) incorporated syntax information into matching; Ji, Lu, and Li (2014) combined several matching features, such as cosine, topic similarity, and translation score, to rank response candidates. In multi-turn conversation, matching requires taking the entire context into consideration. In this scenario, Lowe et al. (2015) used a dual long short-term memory (LSTM) model to match a response with the literal concatenation of utterances in a context; Yan, Song, and Wu (2016) reformulated the input message with the utterances in its previous turns and performed matching with a deep neural network architecture; Zhou et al. (2016) adopted an utterance view and a word view in matching to model relationships among utterances; and Wu et al. (2017) proposed a sequential matching network that can capture important information in contexts and model relationships among utterances in a unified form.

Our work is a retrieval-based method. It is an extension of the work by Wu et al. (2017) reported at the ACL conference. In this work, we analyze the existing models

from a framework view, generalize the model in Wu et al. (2017) to a framework, give another implementation with better performance under the framework, and compare the new model with the model in the conference paper on various aspects.

2.2 Text Matching

In addition to response selection in chatbots, neural network–based text matching techniques have proven effective in capturing semantic relations between text pairs in a variety of NLP tasks. For example, in question answering, covolutional neural networks (Qiu and Huang 2015; Severyn and Moschitti 2015) can effectively capture compositions of n-grams and their relations in questions and answers. Inner-Attention (Wang, Liu, and Zhao 2016) and multiple view (MV)-LSTM (Wan et al. 2016a) can model complex interaction between questions and answers through recurrent neural network based architectures. (More studies on text matching for question answering can be found in Tan et al. [2016]; Liu et al. [2016a,b]; Wan et al. [2016b]; He and Lin [2016]; Yin et al. [2016]; Yin and Schütze [2015]). In Web search, Shen et al. (2014) and Huang et al. (2013) built a neural network with tri-letters to alleviate mismatching of queries and documents due to spelling errors. In textual entailment, the model in Rocktäschel et al. (2015) utilized a word-by-word attention mechanism to distinguish the relationship between two sentences. Wang and Jiang (2016b) introduced another way to adopt an attention mechanism for textual entailment. Besides those two works, Chen et al. (2016), Parikh et al. (2016), and Wang and Jiang (2016a) also investigated the textual entailment problem with neural network models.

In this work, we study text matching for response selection in multi-turn conversation, in which matching is conducted between a piece of text and a context which consists of multiple pieces of text dependent on each other. We propose a new matching framework that is able to extract important information in the context and model dependencies among utterances in the context.

3. Problem Formalization

Suppose that we have a data set $\mathcal{D} = \{(y_i, s_i, r_i)\}_{i=1}^{N}$, where s_i is a conversation context, r_i is a response candidate, and $y_i \in \{0, 1\}$ is a label. $s_i = \{u_{i,1}, \ldots, u_{i,n_i}\}$ where $\{u_{i,k}\}_{k=1}^{n_i}$ are utterances. $\forall k$, $u_{i,k} = (w_{u_{i,k},1}, \ldots, w_{u_{i,k},j}, \ldots, w_{u_{i,k},n_{u_{i,k}}})$ where $w_{u_{i,k},j}$ is the j-th word in $u_{i,k}$ and $n_{u_i,k}$ is the length of $u_{i,k}$. Similarly, $r_i = (w_{r_i,1}, \ldots, w_{r_i,j}, \ldots, w_{r_i,n_{r_i}})$ where $w_{r_i,j}$ is the j-th word in r_i and n_{r_i} is the length of the response. $y_i = 1$ if r_i is a proper response to s_i, otherwise $y_i = 0$. Our goal is to learn a matching model $g(\cdot, \cdot)$ with $\mathcal{D}$, and thus for any new context-response pair (s, r), $g(s, r)$ measures their matching degree. According to $g(s, r)$, we can rank candidates for s and select a proper one as its response.

In the following sections, we first review how the existing work defines $g(\cdot, \cdot)$ from a framework view. The framework view discloses the common drawbacks of the existing work. Then, based on this analysis, we propose a new matching framework and give two models under the framework.

4. A Framework for the Existing Models

Before our work, a few studies on context-response matching for response selection in multi-turn conversation have been conducted. For example, Lowe et al. (2015) match a context and a response with recurrent neural networks (RNNs); Yan et al. (2016) present

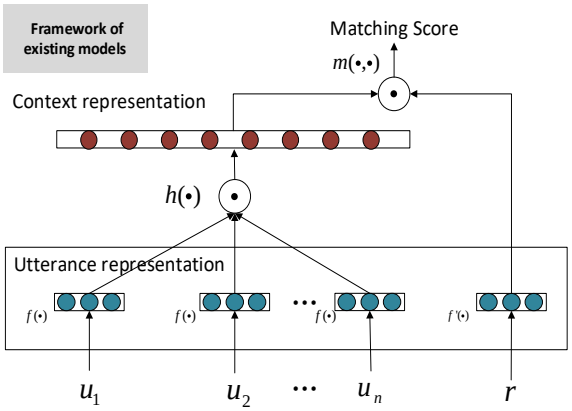


Figure 1
Existing models can be interpreted with a unified framework. $f(\cdot)$, $f'(\cdot)$, $h(\cdot)$, and $m(\cdot,\cdot)$ are utterance representation function, response representation function, context representation function, and matching function, respectively.

a deep learning to respond architecture for multi-turn response selection; and Zhou et al. (2016) perform context-response matching from both a word view and an utterance view. Although these models are proposed from different backgrounds, we find that they can be interpreted with a unified framework, given by Figure 1. The framework consists of utterance representation $f(\cdot)$, response representation $f'(\cdot)$, context representation $h(\cdot)$, and matching calculation $m(\cdot,\cdot)$. Given a context $s = \{u_1, \ldots, u_n\}$ and a response candidate r, $f(\cdot)$ and $f'(\cdot)$ represent each u_i in s and r as vectors or matrices by $f(u_i)$ and $f'(r)$, respectively. $\{f(u_i)\}_{i=1}^{n}$ are then uploaded to $h(\cdot)$, which transforms the utterance representations into $h\left(f(u_1), \ldots, f(u_n)\right)$ as a representation of the context s. Finally, $m(\cdot,\cdot)$ takes $h\left(f(u_1), \ldots, f(u_n)\right)$ and $f'(r)$ as input and calculates a matching score for s and r. To sum up, the framework performs context-response matching following a paradigm that context s and response r are first individually represented as vectors and then their matching degree is determined by the vectors. Under the framework, the matching model $g(s,r)$ can be defined with $f(\cdot)$, $h(\cdot)$, $f'(\cdot)$, and $m(\cdot,\cdot)$, as follows:

$$g(s,r) = m\left(h\left(f(u_1), \ldots, f(u_n)\right), f'(r)\right) \tag{1}$$

The existing models are special cases under the framework with different definitions of $f(\cdot)$, $h(\cdot)$, $f'(\cdot)$, and $m(\cdot,\cdot)$. Specifically, the RNN models in Lowe et al. (2015) can be defined as

$$m_{rnn}(s,r) = \sigma\left(h_{rnn}\left(f_{rnn}(u_1), \ldots, f_{rnn}(u_n)\right)^{\top} \cdot M \cdot f'_{rnn}(r) + b\right) \tag{2}$$

where M is a linear transformation, b is a bias, and $\sigma(\cdot)$ is a sigmoid function. $\forall u_i = \{w_{u_i,1}, \ldots, w_{u_i,n_i}\}$, $f_{rnn}(u_i)$ is defined by

$$f_{rnn}(u_i) = \left[\vec{w}_{u_i,1}, \ldots, \vec{w}_{u_i,k}, \ldots, \vec{w}_{u_i,n_i}\right] \tag{3}$$

where $\vec{w}_{u_i,k}$ is the embedding of the k-th word $w_{u_i,k}$, and $[\cdot]$ denotes a horizontal concatenation operator on vectors or matrices.[1] Suppose that the dimension of the word embedding is d, then the output of $f_{rnn}(u_i)$ is a $d \times n_i$ matrix with each column an embedding vector. Suppose that $r = (w_{r,1}, \ldots, w_{r,n_r})$, then $f'_{rnn}(r)$ is defined as

$$f'_{rnn}(r) = \text{RNN}(\vec{w}_{r,1}, \ldots, \vec{w}_{r,k}, \ldots, \vec{w}_{r,n_r}) \tag{4}$$

where $\vec{w}_{r,k}$ is the embedding of the k-th word in r, and RNN$(\cdot)$ is either a vanilla RNN (Elman 1990) or an RNN with LSTM units (Hochreiter and Schmidhuber 1997). RNN$(\cdot)$ takes a sequence of vectors as an input, and outputs the last hidden state of the network. Finally, the context representation $h_{rnn}(\cdot)$ is defined by

$$h_{rnn}\left(f_{rnn}(u_1), \ldots, f_{rnn}(u_n)\right) = \text{RNN}\left([f_{rnn}(u_1), \ldots, f_{rnn}(u_n)]\right) \tag{5}$$

In the deep learning to respond (DL2R) architecture (Yan, Song, and Wu 2016), the authors first transform the context s to an $s' = \{v_1, \ldots, v_o\}$ with heuristics including "no context," "whole context," "add-one," "drop-out," and "combined." These heuristics differ on how utterances before the last input in the context are incorporated into matching. In "no context," $s' = \{u_n\}$, and thus no previous utterances are considered; in "whole context," $s' = \{u_1 \boxplus \cdots \boxplus u_n, u_n\}$ where operator $\boxplus$ glues vectors together and forms a long vector. Therefore, in "whole context," the conversation context is represented as a concatenation of all its utterances; in "add-one," $s' = \{u_1 \boxplus u_n, \ldots, u_{n-1} \boxplus u_n, u_n\}$. "add-one" leverages the conversation context by concatenating each of its utterances (except the last one) with the last input; in "drop-out," $s' = \{(c \backslash u_1) \boxplus u_n, \ldots, (c \backslash u_{n-1}) \boxplus u_n, u_n\}$ where $c = u_1 \boxplus \cdots \boxplus u_n$ and $c \backslash u_i$ means excluding u_i from c. "drop-out" also utilizes each utterance before the last one individually, but concatenates the complement of each utterance with the last input; and in "combined," s' is the union of the other heuristics. Let $v_o = u_n$ in all heuristics, then the matching model of DL2R can be reformulated as

$$m_{dl2r}(s, r) = \sum_{i=1}^{o} \text{MLP}(f_{dl2r}(v_i) \boxplus f_{dl2r}(v_o)) \cdot \text{MLP}(f_{dl2r}(v_i) \boxplus f'_{dl2r}(r)) \tag{6}$$

where MLP$(\cdot)$ is a multi-layer perceptron. $\forall v \in \{v_1, \ldots, v_o\}$, suppose that $\{\vec{w}_{v,1}, \ldots, \vec{w}_{v,n_v}\}$ represent embedding vectors of the words in v, then $f_{dl2r}(v)$ is given by

$$f_{dl2r}(v) = \text{CNN}\left(\text{Bi-LSTM}(\vec{w}_{v,1}, \ldots, \vec{w}_{v,n_v})\right) \tag{7}$$

where CNN$(\cdot)$ is a convolutional neural network (CNN) (Kim 2014) and Bi-LSTM$(\cdot)$ is a bi-directional recurrent neural network with LSTM units (Bi-LSTM) (Graves, Mohamed, and Hinton 2013). The output of Bi-LSTM$(\cdot)$ is all the hidden states of the Bi-LSTM model. $f'_{dl2r}(\cdot)$ is defined in the same way with $f_{dl2r}(\cdot)$. In DL2R, $h_{dl2r}(\cdot)$ can be viewed as an identity function on $\{f_{dl2r}(v_1), \ldots, f_{dl2r}(v_o)\}$. Note that in the paper of Yan, Song, and Wu (2016), the authors also assume that each response candidate is associated with an antecedent posting p. This assumption does not always hold in multi-turn

1 We borrow the operator from MATLAB.

response selection. For example, in the Ubuntu Dialog Corpus (Lowe et al. 2015), there are no antecedent postings. To make the framework compatible with their assumption, we can simply extend $f_{dl2r}(r)$ to $[f_{dl2r}(p), f_{dl2r}(r)]$, and define $m_{dl2r}(s, r)$ as

$$\sum_{i=1}^{o} \left(\text{MLP}(f_{dl2r}(v_i) \boxplus f_{dl2r}(v_o)) \cdot \left(\sum_{p} \text{MLP}(f_{dl2r}(v_i) \boxplus f_{dl2r}(p)) \cdot \text{MLP}(f_{dl2r}(v_i) \boxplus f_{dl2r}(r)) \right) \right) \quad (8)$$

Finally, in Zhou et al. (2016), the multi-view matching model can be rewritten as

$$m_{mv}(s, r) = \sigma \left(h_{mv}(f_{mv}(u_1), \ldots, f_{mv}(u_n))^\top \begin{bmatrix} M_1 \\ M_2 \end{bmatrix} f'_{mv}(r) + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right) \quad (9)$$

where M_1 and M_2 are linear transformations, b_1 and b_2 are biases. $\forall u_i = \{w_{u_i,1}, \ldots, w_{u_i,n_i}\}$, $f_{mv}(u_i)$ is defined as

$$f_{mv}(u_i) = \{f_w(u_i), f_u(u_i)\} \quad (10)$$

where $f_w(u_i)$ and $f_u(u_i)$ are utterance representations from a word view and an utterance view, respectively. The formulation of $f_w(u_i)$ and $f_u(u_i)$ are given by

$$f_w(u_i) = \left[\vec{w}_{u_i,1}, \ldots, \vec{w}_{u_i,n_i}\right]$$
$$f_u(u_i) = \text{CNN}(\vec{w}_{u_i,1}, \ldots, \vec{w}_{u_i,n_i})$$

Suppose that $r = (w_{r,1}, \ldots, w_{r,n_r})$, then $f'_{mv}(r)$ is defined as

$$f'_{mv}(r) = [f'_w(r)^\top, f'_u(r)^\top]^\top \quad (11)$$

where the word view representation $f'_w(r)$ and the utterance view representation $f'_u(r)$ are formulated as

$$f'_w(r) = \text{GRU}(\vec{w}_{r,1}, \ldots, \vec{w}_{u_{r,n_r}})$$
$$f'_u(r) = \text{CNN}(\vec{w}_{r,1}, \ldots, \vec{w}_{u_{r,n_r}})$$

where GRU$(\cdot)$ is a recurrent neural network with GRUs (Cho et al. 2014). The output of $f'_w(r)$ is the last hidden state of the GRU model. The context representation $h_{mv}(f_{mv}(u_1), \ldots, f_{mv}(u_n))$ is defined as

$$h_{mv}(f_{mv}(u_1), \ldots, f_{mv}(u_n)) = [h_w(f_w(u_1), \ldots, f_w(u_n))^\top, h_u(f_u(u_1), \ldots, f_u(u_n))^\top]^\top \quad (12)$$

where the word view $h_w(\cdot)$ and the utterance view $h_u(\cdot)$ are defined as

$$h_w(f_w(u_1), \ldots, f_w(u_n)) = \text{GRU}\left([f_w(u_1), \ldots, f_w(u_n)]\right)$$
$$h_u(f_u(u_1), \ldots, f_u(u_n)) = \text{GRU}\left(f_u(u_1), \ldots, f_u(u_n)\right)$$

There are several advantages when applying the framework view to the existing context-response matching models. First, it unifies the existing models and reveals the

instinct connections among them. These models are nothing but similarity functions of a context representation and a response representation. Their difference on performance comes from how well the two representations capture the semantics and the structures of the context and the response and how accurate the similarity calculation is. For example, in empirical studies, the multi-view model performs much better than the RNN models. This is because the multi-view model captures the sequential relationship among words, the composition of n-grams, and the sequential relationship of utterances by $h_w(\cdot)$ and $h_u(\cdot)$; whereas in RNN models, only the sequential relationship among words are modeled by $h_{rnn}(\cdot)$. Second, it is easy to make an extension of the existing models by replacing $f(\cdot)$, $f'(\cdot)$, $h(\cdot)$, and $m(\cdot,\cdot)$. For example, we can replace the $h_{rnn}(\cdot)$ in RNN models with a composition of CNN and RNN to model both composition of n-grams and their sequential relationship, and we can replace the $m_{rnn}(\cdot)$ with a more powerful neural tensor network (Socher et al. 2013). Third, the framework unveils the limitations the existing models and their possible extensions suffer: Everything in the context are compressed to one or more fixed-length vectors before matching; and there is no interaction between the context and the response in the formation of their representations. The context is represented without enough supervision from the response, and so is the response. As a result, these models may lose important information of contexts in matching, and more seriously, no matter how we improve them, as long as the improvement is under the framework, we cannot overcome the limitations. The framework view motivates us to propose a new framework that can essentially change the existing matching paradigm.

5. Sequential Matching Framework

We propose a sequential matching framework (SMF) that can simultaneously capture important information in a context and model relationships among utterances in the context. Figure 2 gives the architecture of SMF. SMF consists of utterance–response matching $f(\cdot,\cdot)$, matching accumulation $h(\cdot)$, and matching prediction $m(\cdot)$. The three

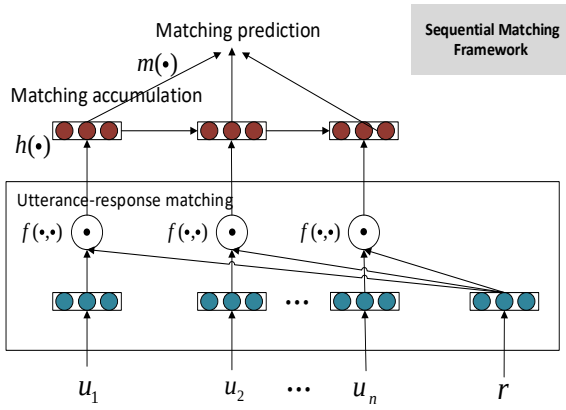


Figure 2
Our new framework for multi-turn response selection, which is called the Sequential Matching Framework. It first computes a matching vector between an utterance and a response, then the matching vectors are accumulated by a GRU. Finally, the matching score is obtained with the hidden states in the second layer.

components are organized in a three-layer architecture. Given a context $s = \{u_1, \ldots, u_n\}$ and a response candidate r, the first layer matches each u_i in s with r through $f(\cdot, \cdot)$ and forms a sequence of matching vectors $\{f(u_1, r), \ldots, f(u_n, r)\}$. Here, we require $f(\cdot, \cdot)$ to be capable of differentiating important parts from unimportant parts in u_i and carry the important information into $f(u_i, r)$. Details of how to design such a $f(\cdot, \cdot)$ will be described later. The matching vectors $\{f(u_1, r), \ldots, f(u_n, r)\}$ are then uploaded to the second layer where $h(\cdot)$ models relationships and dependencies among the utterances $\{u_1, \ldots u_n\}$. Here, we define $h(\cdot)$ as a recurrent neural network whose output is a sequence of hidden states $\{h_1, \ldots, h_n\}$. $\forall k \in \{1, \ldots, n\}$, h_k is given by

$$h_k = h'\Big(h_{k-1}, f(u_k, r)\Big) \tag{13}$$

where $h'(\cdot, \cdot)$ is a non-linear transformation, and $h_0 = 0$. $h(\cdot)$ accumulates matching vectors $\{f(u_1, r), \ldots, f(u_n, r)\}$ in its hidden states. Finally, in the third layer, $m(\cdot)$ takes $\{h_1, \ldots, h_n\}$ as an input and predicts a matching score for (s, r). In brief, SMF matches s and r with a $g(s, r)$ defined as

$$g(s, r) = m\Big(h\Big(f(u_1, r), f(u_2, r), \ldots, f(u_{n_i} r)\Big)\Big) \tag{14}$$

SMF has two major differences over the existing framework: first, SMF lets each utterance in the context and the response "meet" at the very beginning, and therefore utterances and the response can sufficiently interact with each other. Through the interaction, the response will help recognize important information in each utterance. The information is preserved in the matching vectors and carried into the final matching score with minimal loss; second, matching and utterance relationships are coupled rather than separately modeled as in the existing framework. Hence, the utterance relationships (e.g., the order of the utterances), as a kind of knowledge, can supervise the formation of the matching score. Because of the differences, SMF can overcome the drawbacks the existing models suffer and tackle the two challenges of context-response matching simultaneously.

It is obvious that the success of SMF lies in how to design $f(\cdot, \cdot)$, because $f(\cdot, \cdot)$ plays a key role in capturing important information in a context. In the following sections, we will first specify the design of $f(\cdot, \cdot)$, and then discuss how to define $h(\cdot)$ and $m(\cdot)$.

5.1 Utterance–Response Matching

We design the utterance–response matching function $f(\cdot, \cdot)$ in SMF as neural networks to benefit from their powerful representation abilities. To guarantee that $f(\cdot, \cdot)$ can capture important information in utterances with the help of the response, we implement $f(\cdot, \cdot)$ using a convolution-pooling technique and an attention technique, which results in a sequential convolutional network (SCN) and a sequential attention network (SAN). Moreover, in both SCN and SAN, we consider matching on multiple levels of granularity of text. Note that in our ACL paper (Wu et al. 2017), the sequential convolutional network is named "SMN." Here, we rename it to SCN in order to distinguish it from the framework.

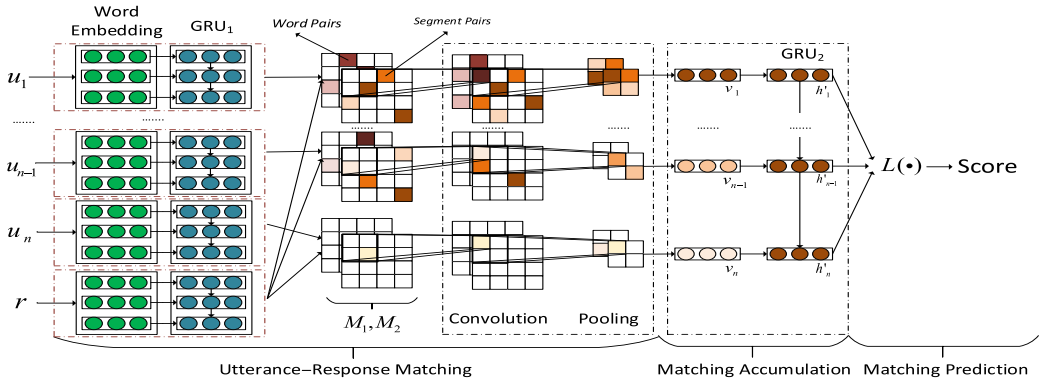


Figure 3
The architecture of SCN. The first layer extracts matching information from interactions between utterances and a response on a word level and a segment level by a CNN. The second layer accumulates the matching information from the first layer by a GRU. The third layer takes the hidden states of the second layer as an input and calculates a matching score.

5.1.1 Sequential Convolutional Network. Figure 3 gives the architecture of SCN. Given an utterance u in a context s and a response candidate r, SCN looks up an embedding table and represents u and r as $\mathbf{U} = [e_{u,1}, \ldots, e_{u,n_u}]$ and $\mathbf{R} = [e_{r,1}, \ldots, e_{r,n_r}]$, respectively, where $e_{u,i}, e_{r,i} \in \mathbb{R}^d$ are the embeddings of the i-th word of u and r, respectively. With $\mathbf{U}$ and $\mathbf{R}$, SCN constructs a word–word similarity matrix $\mathbf{M}_1 \in \mathbb{R}^{n_u \times n_r}$ and a sequence–sequence similarity matrix $\mathbf{M}_2 \in \mathbb{R}^{n_u \times n_r}$ as two input channels of a convolutional neural network (CNN). The CNN then extracts important matching information from the two matrices and encodes the information into a matching vector v.

Specifically, $\forall i, j$, the (i, j)-th element of $\mathbf{M}_1$ is defined by

$$e_{1,i,j} = e_{u,i}^{\top} \cdot e_{r,j} \tag{15}$$

$\mathbf{M}_1$ models the interaction between u and r on a word level.

To get $\mathbf{M}_2$, we first transform $\mathbf{U}$ and $\mathbf{R}$ to sequences of hidden vectors with a GRU. Suppose that $\mathbf{H}_u = [h_{u,1}, \ldots, h_{u,n_u}]$ are the hidden vectors of $\mathbf{U}$, then $\forall i$, $h_{u,i} \in \mathbb{R}^m$ is defined by

$$\begin{aligned} z_i &= \sigma(\mathbf{W_z} e_{u,i} + \mathbf{U_z} h_{u,i-1}) \\ r_i &= \sigma(\mathbf{W_r} e_{u,i} + \mathbf{U_r} h_{u,i-1}) \\ \widetilde{h}_{u,i} &= tanh(\mathbf{W_h} e_{u,i} + \mathbf{U_h}(r_i \odot h_{u,i-1})) \\ h_{u,i} &= z_i \odot \widetilde{h}_{u,i} + (1 - z_i) \odot h_{u,i-1} \end{aligned} \tag{16}$$

where $h_{u,0} = 0$, 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. Similarly, we have $\mathbf{H}_r = [h_{r,1}, \ldots, h_{r,n_r}]$ as the hidden vectors of $\mathbf{R}$. Then, $\forall i, j$, the (i, j)-th element of $\mathbf{M}_2$ is defined by

$$e_{2,i,j} = h_{u,i}^{\top} \mathbf{A} h_{r,j} \tag{17}$$

where $\mathbf{A} \in \mathbb{R}^{m \times m}$ is a linear transformation. $\forall i$, GRU encodes the sequential information and the dependency among words until position i in u into the i-th hidden state. As a consequence, $\mathbf{M}_2$ models the interaction between u and r on a segment level.

$\mathbf{M}_1$ and $\mathbf{M}_2$ are then processed by a CNN to compute the matching vector v. $\forall f = 1, 2$, CNN regards $\mathbf{M}_f$ as an input channel, and alternates convolution and max-pooling operations. If we denote the k-th feature map at the l-th layer as z^k, whose filters are determined by a tensor $\mathbf{W_k}$ and a bias $\mathbf{b_k}$, then the feature map z^k is obtained as follows:

$$z^k_{i,j} = \sigma((\mathbf{W}_k * z')_{i,j} + \mathbf{b}_k) \tag{18}$$

$$z^k_{i,j} = \sigma\left(\left(\sum_u \mathbf{W}^u_k * z'_u\right)_{i,j} + \mathbf{b}_k\right) \tag{19}$$

where $\sigma(\cdot)$ is a ReLU, $\mathbf{W}^u_k$ is the weight of the u-th feature map, $z' = (z'_1 \dots z'_u \dots z'_U)$ is feature maps on the $(l-1)$-th layer, and U is the number of feature maps. Notably, $*$ is a 2D convolutional operation, sliding a window on feature maps at that layer, that is formulated as

$$(\mathbf{W} * o)_{m,n} = \sum_{i=0}^{width} \sum_{j=0}^{height} \mathbf{W}_{i,j} \cdot o_{m+i,n+j} \tag{20}$$

where *width* and *height* are the hyper-parameters of the convolutional window, and $o = z'_u$. A max pooling operation follows a convolution operation and picks the maximal values within a window sliding on the output of the convolution operation, and carries out a linear transformation on the feature values within the window. The max pooling operation can be formulated as

$$z^k_{i,j} = \max z_{(i:i+p_w, j:j+p_h)} \tag{21}$$

where p_w and p_h are the width and the height of the 2D pooling, respectively. The matching vector v is defined by concatenating outputs of the last feature maps and transforming it to a low dimensional space:

$$v = \mathbf{W_c}[z^0, z^1 \dots, z^{f'}] + \mathbf{b_c} \tag{22}$$

where f' denotes the number of feature maps, $\mathbf{W_c}$ and $\mathbf{b_c}$ are parameters, and z^k is the concatenation of elements at the k-th feature map, meaning $z^k = [z^k_{0,0}, z^k_{0,1} \dots z^k_{I,J}]$ where I and J are the maximum indices of the feature map.

SCN distills important information in each utterance in the context from multiple levels of granularity through convolution and pooling operations on similarity matrices. From Equations (15), (17), (18), and (21), we can see that by learning word embeddings and parameters of GRU from training data, important words or segments in the utterance may have high similarity with some words or segments in the response and result in high value areas in the similarity matrices. These areas will be transformed and extracted to the matching vector by convolutions and poolings. We will further explore the mechanism of SCN by visualizing $\mathbf{M}_1$ and $\mathbf{M}_2$ of an example in Section 7.

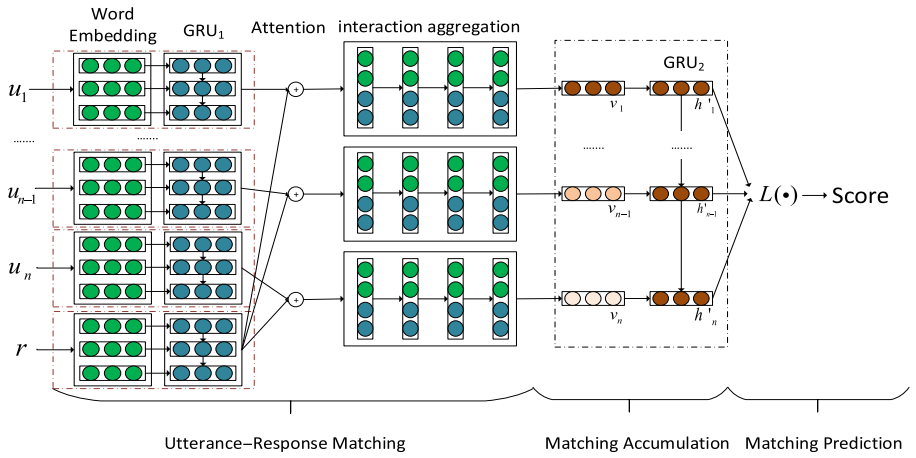


Figure 4
The architecture of SAN. The first layer highlights important words and segments in context, and computes a matching vector from both word level and segment level. Similar to SCN, the second layer uses a GRU to accumulate the matching information, and the third layer predicts the final matching score.

5.1.2 Sequential Attention Network. With word embeddings **U** and **R** and hidden vectors $\mathbf{H}_u$ and $\mathbf{H}_r$, SAN also performs utterance–response matching on a word level and a segment level. Figure 4 gives the architecture of SAN. In each level of matching, SAN exploits every part of the response (either a word or a hidden state) to weight the parts of the utterance and obtain a weighted representation of the utterance. The utterance representation then interacts with the part of the response. The interactions are finally aggregated following the order of the parts in the response as a matching vector.

Specifically, $\forall e_{r,i} \in \mathbf{R}$, the weight of $e_{u,j} \in \mathbf{U}$ is given by

$$\omega_{i,j} = tanh(e_{u,j}^{\top} \mathbf{W}_{\mathbf{att1}} e_{r,i} + \mathbf{b}_{\mathbf{att1}}) \tag{23}$$

$$\alpha_{i,j} = \frac{e^{\omega_{i,j}}}{\sum_{j=1}^{n_u} e^{\omega_{i,j}}} \tag{24}$$

where $\mathbf{W}_{\mathbf{att1}} \in \mathbb{R}^{d \times d}$, and $\mathbf{b}_{\mathbf{att1}} \in \mathbb{R}$ are parameters. $\omega_{i,j} \in \mathbb{R}$ represents the importance of $e_{u,j}$ in the utterance corresponding to $e_{r,i}$ in the response. $\alpha_{i,j}$ is normalized importance. The interaction between u and $e_{r,i}$ is then defined as

$$t_{1,i} = \left(\sum_{j=1}^{n_u} \alpha_{i,j} e_{u,j} \right) \odot e_{r,i} \tag{25}$$

where $(\sum_{j=1}^{n_u} \alpha_{i,j} e_{u,j})$ is the representation of u with weights $\{\alpha_{i,j}\}_{j=1}^{n_u}$, and $\odot$ is the Hadamard product.

Similarly, $\forall h_{r,i} \in \mathbf{H}_r$, the weight of $h_{u,j} \in \mathbf{H}_u$ can be defined as

$$\omega'_{i,j} = v'^{\top} tanh(h_{u,j}^{\top} \mathbf{W_{att2}} h_{r,i} + \mathbf{b_{att2}}) \tag{26}$$

$$\alpha'_{i,j} = \frac{e^{\omega'_{i,j}}}{\sum_{j=1}^{n_u} e^{\omega'_{i,j}}} \tag{27}$$

where $\mathbf{W_{att2}} \in \mathbb{R}^{d \times d}$, $v' \in \mathbb{R}^d$, and $\mathbf{b_{att2}} \in \mathbb{R}^d$ are parameters. The interaction between u and $h_{r,i}$ then can be formulated as

$$t_{2,i} = \sum_{j=1}^{n_u} \left(\alpha'_{i,j} h_{u,j} \right) \odot h_{r,i} \tag{28}$$

We denote the attention weights $\{\alpha_{i,j}\}$ and $\{\alpha'_{i,j}\}$ as $\mathbf{A_1}$ and $\mathbf{A_2}$, respectively. With the word-level interaction $\mathbf{T}_1 = [t_{1,1}, \ldots, t_{1,n_r}]$ and the segment level interaction $\mathbf{T}_2 = [t_{2,1}, \ldots, t_{2,n_r}]$, we form a $\mathbf{T} = [t_1, \ldots, t_{n_r}]$ by defining t_i as $[t_{1,i}^{\top}, t_{2,i}^{\top}]^{\top}$. The matching vector v of SAN is then obtained by processing $\mathbf{T}$ with a GRU:

$$v = \text{GRU}(\mathbf{T}) \tag{29}$$

where the specific parameterization of GRU$(\cdot)$ is similar to Equation (16), and we take the last hidden state of the GRU as v.

From Equations (23) and (26), we can see that SAN identifies important information in utterances in a context through an attention mechanism. Words or segments in utterances that are useful to recognize the appropriateness between the context and a response will receive high weights from the response. The information conveyed by these words and segments will be highlighted in the interaction between the utterances and the response and carried to the matching vector through a RNN that models the aggregation of information in the utterances under the supervision of the response. Similar to SCN, we will further investigate the effect of the attention mechanism in SAN by visualizing the attention weights in Section 7.

5.1.3 SAN vs. SCN. Because SCN and SAN exploits different mechanisms to understand important parts in contexts, an interesting question arises: What are the advantages and disadvantages of the two models in practice? Here, we leave empirical comparison of their performance to experiments and first compare SCN with SAN on the following aspects: (1) amount of parallelable computation, which is measured by the minimum number of sequential operations required; and (2) total time complexity.

Table 2 summarizes the comparison between the two models. In terms of parallelability, SAN uses two RNNs to learn the representations, which requires $2n$ sequential operations, whereas SCN has n sequentially executed operations in the construction of $\mathbf{M}_2$. Hence, SCN is easier to parallelize than SAN. In terms of time complexity, the complexity of SCN is $\mathcal{O}(k \cdot n \cdot d^2 + n \cdot d^2 + n^2 \cdot d)$, where k is the number of feature maps in convolutions, n is $max(n_u, n_r)$, and d is embedding size. More specifically, in SCN, the cost on construction of $\mathbf{M}_1$ and $\mathbf{M}_2$ is $\mathcal{O}(n \cdot d^2 + n^2 \cdot d)$, and the cost on convolution and pooling is $\mathcal{O}(k \cdot n \cdot d^2)$. The complexity of SAN is $\mathcal{O}(n^2 \cdot d + n^2 \cdot d^2)$, where $\mathcal{O}(n^2 \cdot d)$ is the cost on calculating $\mathbf{H}_u$ and $\mathbf{H}_r$ and $\mathcal{O}(n^2 \cdot d^2)$ is the cost of the following attention-based GRU. In practice, k is usually much smaller than the maximum sentence length n.

Table 2
Comparison between SCN and SAN. k is the kernel number of convolutions. n is $max(n_u, n_r)$. d is the embedding size.

	time complexity	number of sequential operations
SCN	$\mathcal{O}(k \cdot n \cdot d^2 + n \cdot d^2 + n^2 \cdot d)$	n
SAN	$\mathcal{O}(n^2 \cdot d^2 + n^2 \cdot d)$	$2n$

Therefore, SCN could be faster than SAN. The conclusion is also verified by empirical results in Section 7.

5.2 Matching Accumulation

The function of matching accumulation $h(\cdot)$ in SMF can be implemented with any recurrent neural networks such as LSTM and GRU. In this work, we fix $h(\cdot)$ as GRU in both SCN and SAN. Given $\{f(u_1, r), \ldots, f(u_n, r)\}$ as the output of the first layer of SMF, the non-linear transformation $h'(\cdot, \cdot)$ in Equation (13) is formulated as

$$\begin{aligned} z'_i &= \sigma(\mathbf{W_z}' f(u_i, r) + \mathbf{U_z}' h_{i-1}) \\ r'_i &= \sigma(\mathbf{W_r}' f(u_i, r) + \mathbf{U_r}' h_{i-1}) \\ \widetilde{h}_i &= tanh(\mathbf{W_h}' f(u_i, r) + \mathbf{U_h}' (r_i \odot h'_{i-1})) \\ h_i &= z_i \odot \widetilde{h}_i + (1 - z_i) \odot h_{i-1} \end{aligned} \tag{30}$$

where $\mathbf{W_z}'$, $\mathbf{W_h}'$, $\mathbf{W_r}'$, $\mathbf{U_z}'$, $\mathbf{U_r}'$,$\mathbf{U_h}'$ are parameters, and z'_i and r'_i are an update gate and a reset gate, respectively. Here, h_i is a hidden state, which encodes the matching information in its previous turns. From Equation (30) we can see that the reset gate (i.e., r_i) and the update gate (i.e., z_i) control how much information from the current matching vector $f(u_i, r)$ flows into the accumulation vector h_i. Ideally, the two gates should let matching vectors that correspond to important utterances make a great impact to the accumulation vectors (i.e., the hidden states) while blocking the information from the unimportant utterances. In practice, we find that we can achieve this by learning SCN and SAN from large-scale conversation data. The details will be given in Section 7.

5.3 Matching Prediction

$m(\cdot)$ takes $\{h_1, \ldots, h_n\}$ from $h(\cdot)$ as an input and predicts a matching score for (s, r). We consider three approaches to implementing $m(\cdot)$.

5.3.1 Last State. The first approach is that we only use the last hidden state h_n to calculate a matching score. The underlying assumption is that important information in the context, after selection by the gates of the GRU, has been encoded into the vector h_n. Then $m(\cdot)$ is formulated as

$$m_{last}(h_1, \ldots, h_n) = softmax(\mathbf{W_l} h_n + \mathbf{b_l}) \tag{31}$$

where $\mathbf{W_l}$ and $\mathbf{b_l}$ are parameters.

5.3.2 Static Average. The second approach is combining all hidden states with weights determined by their positions. In this approach, $m(\cdot)$ can be formulated as

$$m_{static}(h_1, \ldots, h_n) = softmax(\mathbf{W_s}(\sum_{i=1}^{n} w_i h_i) + \mathbf{b_s}) \tag{32}$$

where $\mathbf{W_s}$ and $\mathbf{b_s}$ are parameters, and w_i is the weight of the i-th hidden state and is learned from data. Note that in $m_{static}(\cdot)$, once $\{w_i\}_{i=1}^{n}$ are learned, they are fixed for any (s, r) pairs, and that is why we call the approach "static average." Compared with last state, the static average can leverage more information in the early parts of $\{h_1, \ldots, h_n\}$, and thus can avoid information loss from the process of the GRU in $h(\cdot)$.

5.3.3 Dynamic Average. Similar to static average, we also combine all hidden states to calculate a matching score, but the difference is that the combination weights are dynamically computed by the hidden states and the utterance vectors through an attention mechanism as in Bahdanau, Cho, and Bengio (2014). The weights will change according to the content of the utterances in different contexts, and that is why we call the approach "dynamic average." In this approach, $m(\cdot)$ is defined as

$$\begin{aligned} t_i &= t_s^\top tanh(\mathbf{W_{d1}} h_{u,n_u} + \mathbf{W_{d2}} h_i + \mathbf{b_{d1}}) \\ \alpha_i &= \frac{exp(t_i)}{\sum_i exp(t_i)} \\ m(h_1, \ldots, h_n) &= softmax(\mathbf{W_d}(\sum_{i=1}^{n} \alpha_i h_i) + \mathbf{b_{d2}}) \end{aligned} \tag{33}$$

where $\mathbf{W_{d1}} \in \mathbb{R}^{q \times m}$, $\mathbf{W_{d2}} \in \mathbb{R}^{q \times q}$, $\mathbf{b_{d1}} \in \mathbb{R}^q$, $\mathbf{W_d} \in \mathbb{R}^{q \times q}$, and $\mathbf{b_{d2}} \in \mathbb{R}^q$ are parameters. t_s is a virtual context vector that is learned in training. h_i and h_{u,n_u} are i-th hidden state of $h(\cdot)$ and the final hidden state of the utterance, respectively.

6. Model Training

We choose cross entropy as the loss function. Let Θ denote the parameters of $f(\cdot, \cdot)$, $h(\cdot, \cdot)$, and $m(\cdot)$, then the objective function $\mathcal{L}(\mathcal{D}, \Theta)$ can be written as

$$\mathcal{L}(\mathcal{D}, \Theta) = -\sum_{i=1}^{N} \left[y_i log(g(s_i, r_i)) + (1 - y_i) log(1 - g(s_i, r_i)) \right] \tag{34}$$

where N in the number of instances in $\mathcal{D}$. We optimize the objective function using back-propagation and the parameters are updated by stochastic gradient descent with the Adam algorithm (Kingma and Ba 2014) on a single Tesla K80 GPU. The initial learning rate is 0.001, and the parameters of Adam, β_1 and β_2, are 0.9 and 0.999, respectively. We use early-stopping as a regularization strategy. Models are trained in mini-batches with a batch size of 200.

7. Experiments

We test SAN and SCN on two public data sets with both quantitative metrics and qualitative analysis.

7.1 Data Sets

The first data set we exploited to test the performance of our models is the Ubuntu Dialogue Corpus v1 (Lowe et al. 2015). The corpus contains large-scale two-way conversations collected from the chat logs of the Ubuntu forum. The conversations are multi-turn discussions about Ubuntu-related technical issues. We used the copy shared by Xu et al. (Xu et al. 2017),[2] in which numbers, URLs, and paths are replaced by special placeholders. The data set consists of 1 million context-response pairs for training, 0.5 million pairs for validation, and 0.5 million pairs for testing. In each conversation, a human reply is selected as a positive response to the context, and negative responses are randomly sampled. The ratio of positive responses and negative responses is 1:1 in the training set, and 1:9 in both the validation and test sets.

In addition to the Ubuntu Dialogue Corpus, we selected the Douban Conversation Corpus (Wu et al. 2017) as another data set. The data set is a recently released large-scale open-domain conversation corpus in which conversations are crawled from a popular Chinese forum Douban Group.[3] The training set contains 1 million context-response pairs, and the validation set contains 50,000 pairs. In both sets, a context has a human reply as a positive response and a randomly sampled reply as a negative response. Therefore, the ratio of positive instances and negative instances in both training and validation is 1:1. Different from the Ubuntu Dialogue Corpus, the test set of the Douban Conversation Corpus contains 1,000 contexts with each one having 10 responses retrieved from a pre-built index. Each response receives three labels from human annotators that indicate its appropriateness as a reply to the context and the majority of the labels are taken as the final decision. An appropriate response means that the response can naturally reply to the conversation history by satisfying logic consistency, fluency, and semantic relevance. Otherwise, if a response does not meet any of the three conditions, it is an inappropriate response. The Fleiss kappa (Fleiss 1971) of the labeling is 0.41, which means that the labelers reached a moderate agreement in their work. Note that in our experiments, we removed contexts whose responses are all labeled as positive or negative. After this step, there are 6,670 context-response pairs left in the test set.

Table 3 summarizes the statistics of the two data sets.

7.2 Baselines

We compared our methods with the following methods:

TF-IDF: We followed Lowe et al. (2015) and computed TF-IDF-based cosine similarity between a context and a response. Utterances in the context are concatenated to form a document. IDF is computed on the training data.

Basic deep learning models: We used models in Lowe et al. (2015) and Kadlec, Schmid, and Kleindienst (2015), in which representations of a context are learned by

2 https://www.dropbox.com/s/2fdn26rj6h9bpvl/ubuntudata.zip?dl=0.
3 https://www.douban.com/group/.

Table 3
Statistics of the two data sets.

	Ubuntu Corpus			Douban Corpus		
	train	val	test	train	val	test
# context-response pairs	1M	0.5M	0.5M	1M	50k	10k
# candidates per context	2	10	10	2	2	10
# positive candidates per context	1	1	1	1	1	1.18
Min. # turns per context	3	3	3	3	3	3
Max. # turns per context	19	19	19	98	91	45
Avg. # turns per context	10.10	10.10	10.11	6.69	6.75	6.45
Avg. # words per utterance	12.45	12.44	12.48	18.56	18.50	20.74

neural networks with the concatenation of utterances as inputs and the final matching score is computed by a bilinear function of the context representation and the response representation. Models including RNN, CNN, LSTM, and BiLSTM were selected as baselines.

Multi-View: The model proposed in Zhou et al. (2016) that utilizes a hierarchical recurrent neural network to model utterance relationships. It integrates information in a context from an utterance view and a word view. Details of the model can be found in Equation (9).

Deep learning to respond (DL2R): The authors in Yan, Song, and Wu (2016) proposed several approaches to reformulate a message with previous turns in a context. The response and the reformulated message are then represented by a composition of RNN and CNN. Finally, the matching score is computed with the concatenation of the representations. Details of the model can be found in Equation (6).

Advanced single-turn matching models: Because BiLSTM does not represent the state-of-the-art matching model, we concatenated the utterances in a context and matched the long text with a response candidate using more powerful models, including MV-LSTM (Wan et al. 2016b) (2D matching), Match-LSTM (Wang and Jiang 2016b), and Attentive-LSTM (Tan et al. 2016) (two attention based models). To demonstrate the importance of modeling utterance relationships, we also calculated a matching score for the concatenation of utterances and the response candidate using the methods in Section 5.1. The two models are simple versions of SCN and SAN, respectively, without considering utterance relationships. We denote them as SCN_{single} and SAN_{single}, respectively.

7.3 Evaluation Metrics

In experiments on the Ubuntu corpus, we followed Lowe et al. (2015) and used recall at position k in n candidates ($R_n@k$) as evaluation metrics. Here the matching models are required to return k most likely responses, and $R_n@k = 1$ if the true response is among the k candidates. $R_n@k$ will become larger when k gets larger or n gets smaller.

$R_n@k$ has bias when there are multiple true candidates for a context. Hence, on the Douban corpus, apart from $R_n@k$s, we also followed the convention of information retrieval and used mean average precision (MAP) (Baeza-Yates, Ribeiro-Neto et al.

1999), mean reciprocal rank (MRR) (Voorhees and Tice 2000), and precision at position 1 (P@1) as evaluation metrics, which are defined as follows

$$MAP = \frac{1}{|S|}\sum_{s_i \in S} AP(s_i) \text{ , where } AP(s_i) = \frac{\sum_{j=0}^{N_r} \frac{\sum_{k=0}^{j} rel(r_k, s_i)}{j} \cdot rel(r_j, s_i)}{\sum_{j=0}^{N_r} rel(r_j, s_i)} \tag{35}$$

$$MRR = \frac{1}{|S|}\sum_{s_i \in S} RR(s_i) \text{ , where } RR(s_i) = \frac{1}{rank_i} \tag{36}$$

$$P@1 = \frac{1}{|S|}\sum_{s_i \in S} rel(r_{top1}, s_i) \tag{37}$$

where $rank_i$ refers to the position of the first relevant response to context s_i in the ranking list; r_j refers to the response ranked at the j-th position; $rel(r_j, s_i) = 1$ if r_j is an appropriate response to context s_i, otherwise $rel(r_j, s_i) = 0$; r_{top1} is the response ranked at the top position; S is the universal set of contexts; and N_r denotes the number of retrieved responses.

We did not calculate $R_2@1$ for the test data in the Douban corpus because one context could have more than one correct response, and we have to randomly sample one for $R_2@1$, which may bring bias to the evaluation.

7.4 Parameter Tuning

For baseline models, we copied the numbers in the existing papers if their results on the Ubuntu corpus are reported in their original paper (TF-IDF, RNN, CNN, LSTM, BiLSTM, Multi-View); otherwise we implemented the models by tuning their parameters on the validation sets. All models were implemented using the Theano framework (Theano Development Team 2016). Word embeddings in neural networks were initialized by the results of word2vec (Mikolov et al. 2013[4]) pre-trained on the training data. We did not use GloVe (Pennington, Socher, and Manning 2014) because the Ubuntu corpus contains many technical words that are not covered by Twitter or Wikipedia. The word embedding size was chosen as 200. The maximum utterance length was set as 50. The maximum context length (i.e., number of utterances per context) was varied from 1 to 20 and set as 10 at last. We padded zeros if the number of utterances in a context is less than 10; otherwise we kept the last 10 utterances. We will discuss how the performance of models changes in terms of different maximum context length later.

For SCN, the window size of convolution and pooling was tuned to $\{(2,2), (3,3)(4,4)\}$ and was set as $(3,3)$ finally. The number of feature maps is 8. The size of the hidden states in the construction of $\mathbf{M}_2$ is the same with the word embedding size, and the size of the output vector v was set as 50. Furthermore, the size of the hidden states in the matching accumulation module is also 50. In SAN, the size of the hidden states in the segment level representation is 200, and the size of the hidden states in Equation (29) was set as 400.

All tuning was done according to $R_2@1$ on the validation data.

4 https://code.google.com/archive/p/word2vec/.

7.5 Evaluation Results

Tables 4 and 5 show the evaluation results on the Ubuntu Corpus and the Douban Corpus, respectively. SAN and SCN outperform baselines over all metrics on both data sets with large margins, and except for R_{10}@5 of SCN on the Douban corpus, the improvements are statistically significant (t-test with p-value ≤ 0.01). Our models are better than state-of-the-art single turn matching models such as MV-LSTM, Match-LSTM, SCN_{single}, and SAN_{single}. The results demonstrate that one cannot neglect utterance relationships and simply perform multi-turn response selection by concatenating utterances together.

TF-IDF shows the worst performance, indicating that the multi-turn response selection problem cannot be addressed with shallow features. LSTM is the best model among the basic models. The reason might be that it models relationships among words. Multi-View is better than LSTM, demonstrating the effectiveness of the utterance-view in context modeling. Advanced models have better performance, because they are capable of capturing more complicated structures in contexts.

SAN is better than SCN on both data sets, which might be attributed to three reasons. The first reason is that SAN uses vectors instead of scalars to represent interactions between words or text segments. Therefore, the matching vectors in SAN can encode more information from the pairs than those in SCN. The second reason is that SAN uses a soft attention mechanism to emphasize important words or segments

Table 4
Evaluation results on the Ubuntu corpus. Subscripts including last, static, and dynamic indicate three approaches to predicting a matching score as described in Section 5.3. Numbers in **bold** mean that the improvement from the models is statistically significant over the best baseline method.

	R_2@1	R_{10}@1	R_{10}@2	R_{10}@5
TF-IDF	0.659	0.410	0.545	0.708
RNN	0.768	0.403	0.547	0.819
CNN	0.848	0.549	0.684	0.896
LSTM	0.901	0.638	0.784	0.949
BiLSTM	0.895	0.630	0.780	0.944
Multi-View	0.908	0.662	0.801	0.951
DL2R	0.899	0.626	0.783	0.944
MV-LSTM	0.906	0.653	0.804	0.946
Match-LSTM	0.904	0.653	0.799	0.944
Attentive-LSTM	0.903	0.633	0.789	0.943
SCN_{single}	0.904	0.656	0.809	0.942
SAN_{single}	0.906	0.662	0.810	0.945
SCN_{last}	**0.923**	**0.723**	**0.842**	**0.956**
SCN_{static}	**0.927**	**0.725**	**0.838**	**0.962**
$SCN_{dynamic}$	**0.926**	**0.726**	**0.847**	**0.961**
SAN_{last}	**0.930**	**0.733**	**0.850**	**0.961**
SAN_{static}	**0.932**	**0.734**	**0.852**	**0.962**
$SAN_{dynamic}$	**0.932**	**0.733**	**0.851**	**0.961**

Table 5
Evaluation results on the Douban corpus. Notations have the same meaning as those in Table 4. On R_{10}@5, only SAN significantly outperforms baseline methods.

	MAP	MRR	P@1	R_{10}@1	R_{10}@2	R_{10}@5
TF-IDF	0.331	0.359	0.180	0.096	0.172	0.405
RNN	0.390	0.422	0.208	0.118	0.223	0.589
CNN	0.417	0.440	0.226	0.121	0.252	0.647
LSTM	0.485	0.527	0.320	0.187	0.343	0.720
BiLSTM	0.479	0.514	0.313	0.184	0.330	0.716
Multi-View	0.505	0.543	0.342	0.202	0.350	0.729
DL2R	0.488	0.527	0.330	0.193	0.342	0.705
MV-LSTM	0.498	0.538	0.348	0.202	0.351	0.710
Match-LSTM	0.500	0.537	0.345	0.202	0.348	0.720
Attentive-LSTM	0.495	0.523	0.331	0.192	0.328	0.718
SCN_{single}	0.506	0.543	0.349	0.203	0.351	0.709
SAN_{single}	0.508	0.547	0.352	0.206	0.353	0.720
SCN_{last}	**0.526**	**0.571**	**0.393**	**0.236**	**0.387**	0.729
SCN_{static}	**0.523**	**0.572**	**0.387**	**0.228**	**0.387**	0.734
$SCN_{dynamic}$	**0.529**	**0.569**	**0.397**	**0.233**	**0.396**	0.724
SAN_{last}	**0.536**	**0.581**	**0.393**	**0.236**	**0.404**	**0.761**
SAN_{static}	**0.532**	**0.575**	**0.387**	**0.228**	**0.393**	**0.736**
$SAN_{dynamic}$	**0.534**	**0.577**	**0.391**	**0.230**	**0.393**	**0.742**

in utterances, whereas SCN uses a max pooling operation to select important information from similarity matrices. When multiple words or segments are important in an utterance–response pair, a max pooling operation just selects the top one, but the attention mechanism can leverage all of them. The last reason is that SAN models the sequential relationship and dependency among words or segments in the interaction aggregation module, whereas SCN only considers n-grams.

The three approaches to matching prediction do not show much difference in both SCN and SAN, but dynamic average and static average are better than the last state on the Ubuntu corpus and worse than it on the Douban corpus. This is because contexts in the Ubuntu corpus are longer than those in the Douban corpus (average context length 10.1 vs. 6.7), and thus the last hidden state may lose information in history on the Ubuntu data. In contrast, the Douban corpus has shorter contexts but longer utterances (average utterance length 18.5 vs. 12.4), and thus noise may be involved in response selection if more hidden states are taken into consideration.

There are two reasons that R_n@ks on the Douban corpus are much smaller than those on the Ubuntu corpus. One is that response candidates in the Douban corpus are returned by a search engine rather than negative sampling. Therefore, some negative responses in the Douban corpus might be semantically closer to the true positive responses than those in the Ubuntu corpus, and thus more difficult to differentiate by a model. The other is that there are multiple correct candidates for a context, so the maximum R_{10}@1 for some contexts are not 1. For example, if there are three correct responses, then the maximum R_{10}@1 is 0.33. P@1 is about 40% on the Douban corpus, indicating the difficulty of the task in a real chatbot.

7.6 Further Analysis

7.6.1 Model Ablation. We first investigated how different parts of SCN and SAN affect their performance by ablating SCN_{last} and SAN_{last}. Table 6 reports the results of ablation on the test data. First, we replaced the utterance–response matching module in SCN and SAN with a neural tensor (Socher et al. 2013) (denoted as $Replace_M$), which matches an utterance and a response by feeding their representations to a neural tensor network (NTN). The result is that the performance of the two models dropped dramatically. This is because in NTN there is no interaction between the utterance and the response before their matching; and it is doubtful whether NTN can recognize important parts in the pair and encode the information into matching. As a result, the model loses important information in the pair. Therefore, we can conclude that a good utterance–response matching mechanism is crucial to the success of SMF. At least, one has to let an utterance and a response interact with each other and explicitly highlight important parts in their matching vector. Second, we replaced the GRU in the matching accumulation modules of SCN and SAN with a multi-layer perceptron (denoted as SCN $Replace_A$ and SAN $Replace_A$, respectively). The change led to a slight performance drop. This indicates that utterance relationships are useful in context-response matching. Finally, we only left one level of granularity, either word level or segment level, in SCN and SAN, and denoted the models as SCN with words, SCN with segments, SAN with words, and SAN with segments, respectively. The results indicate that segment level matching on utterance–response pairs contributes more to the final context-response matching, and both segments and words are useful in response selection.

7.6.2 Comparison with Respect to Context Length. We then studied how the performance of SCN_{last} and SAN_{last} changes across contexts with different lengths. Context-response pairs were bucketed into three bins according to the length of the contexts (i.e., the number of utterances in the contexts), and comparison was made in different bins on different metrics. Figure 5 gives the results. Note that we did the analysis only on the Douban corpus because on the Ubuntu corpus many results were copied from the existing literatures and the bin-level results are not available. SAN and SCN consistently perform better than the baselines over bins, and a general trend is that when contexts become longer, gaps become larger. For example, in $(2, 5]$, SAN is 3 points higher than LSTM on $R_{10}@5$, but the gap becomes 5 points in $(5, 10]$. The results demonstrate that our models can well capture dependencies, especially long-distance dependencies, among utterances in contexts. SAN and SCN have similar trends because both of them use a

Table 6
Evaluation results of model ablation.

	Ubuntu Corpus				Douban Corpus					
	$R_2@1$	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$	MAP	MRR	P@1	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$
$Replace_M$	0.905	0.661	0.799	0.950	0.503	0.541	0.343	0.201	0.364	0.729
SCN with words	0.919	0.704	0.832	0.955	0.518	0.562	0.370	0.228	0.371	0.737
SCN with segments	0.921	0.715	0.836	0.956	0.521	0.565	0.382	0.232	0.380	0.734
SCN $Replace_A$	0.918	0.716	0.832	0.954	0.522	0.565	0.376	0.220	0.385	0.727
SCN_{last}	0.923	0.723	0.842	0.956	0.526	0.571	0.393	0.236	0.387	0.729
SAN with words	0.922	0.713	0.842	0.957	0.523	0.565	0.372	0.232	0.381	0.747
SAN with segments	0.928	0.729	0.846	0.959	0.532	0.575	0.385	0.234	0.393	0.754
SAN $Replace_A$	0.927	0.728	0.842	0.959	0.532	0.561	0.386	0.225	0.395	0.757
SAN_{last}	0.930	0.733	0.850	0.961	0.536	0.581	0.393	0.236	0.404	0.761

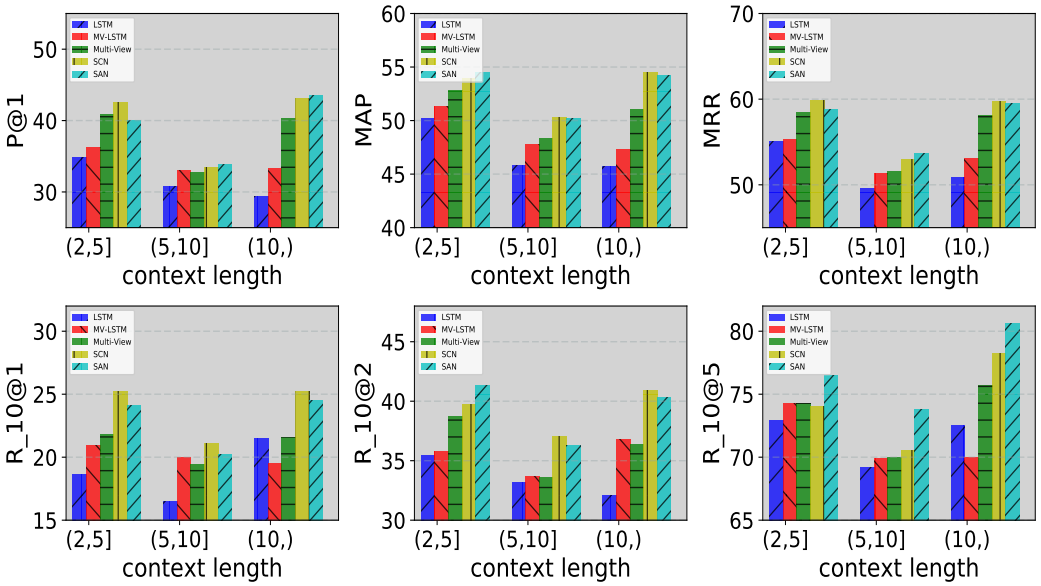


Figure 5
Model performance across context length. We compared SAN and SCN with LSTM, MV-LSTM, and Multi-View on the Douban corpus.

GRU in the second layer to model dependencies among utterances. The performance of all models drops when the length of contexts increases from (2, 5] to (5, 10]. This is because semantics of longer contexts is more difficult to capture than that of shorter contexts. On the other hand, the performance of all models improved when the length of contexts increases from (5, 10] to (10,). This is because the bin of (10,) contains much less data than the other two bins (the data distribution is 53% for (2, 5], 38% for (5, 10], and 9% for (10,)), and thus the improvement does not make much sense from a statistical perspective.

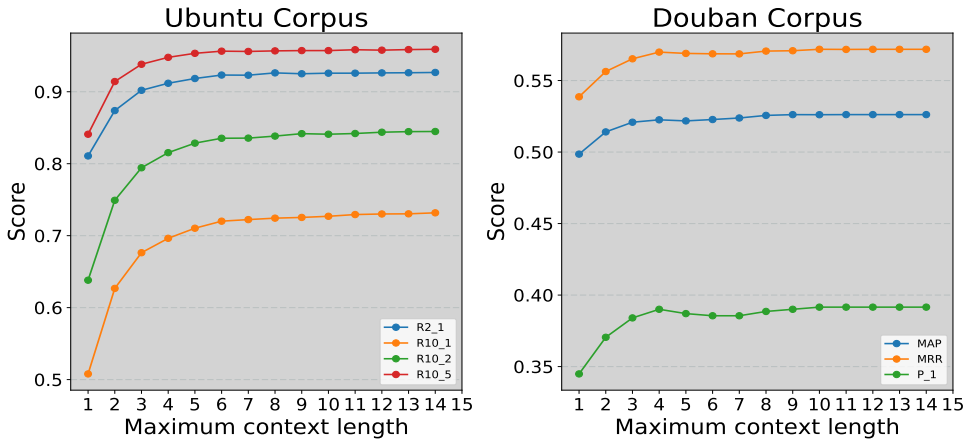
7.6.3 Sensitivity to Hyper-Parameters. We checked how sensitive SCN and SAN are regarding the size of word embedding and the maximum context length. Table 7 reports evaluation results of SCN_{last} and SAN_{last} with embedding sizes varying in {50, 100, 200}.

Table 7
Evaluation results in terms of different word embedding sizes.

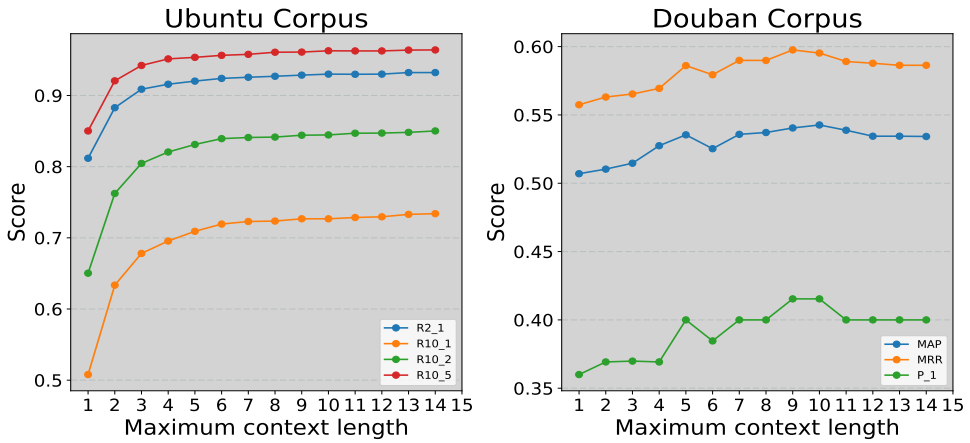
	Ubuntu Corpus				Douban Corpus					
	R_2@1	R_{10}@1	R_{10}@2	R_{10}@5	MAP	MRR	P@1	R_{10}@1	R_{10}@2	R_{10}@5
SCN_{50d}	0.920	0.715	0.834	0.952	0.503	0.541	0.343	0.201	0.364	0.729
SCN_{100d}	0.921	0.718	0.838	0.954	0.524	0.569	0.391	0.234	0.387	0.727
SCN_{200d}	0.923	0.723	0.842	0.956	0.526	0.571	0.393	0.236	0.387	0.729
SAN_{50d}	0.914	0.698	0.828	0.950	0.503	0.541	0.343	0.201	0.364	0.729
SAN_{100d}	0.921	0.711	0.840	0.953	0.525	0.565	0.375	0.220	0.388	0.746
SAN_{200d}	0.930	0.733	0.850	0.961	0.536	0.581	0.393	0.236	0.404	0.761

We can see that SAN is more sensitive to the word embedding size than SCN. SCN becomes stable after the embedding size exceeds 100, whereas SAN keeps improving with the increase of the embedding size. Our explanation of the phenomenon is that SCN transforms word vectors and hidden vectors of GRU to scalars in the similarity matrices by dot products, thus information in extra dimensions (e.g., entries with indices larger than 100) might be lost; on the other hand, SAN leverages the whole d-dimensional vectors in matching, so the information in the embedding can be exploited more sufficiently.

Figure 6 shows the performance of SCN and SAN with respect to the maximum context length. We find that both models significantly become better with the increase of maximum context length when it is lower than 5, and become stable after the maximum context length reaches 10. The results indicate that utterances from early history can provide useful information to response selection. Moreover, model performance is more sensitive to the maximum context length on the Ubuntu corpus than it is on the Douban



(a) Performance of SCN across different context lengths.



(b) Performance of SAN across different context lengths.

Figure 6
Performance with respect to different maximum context lengths.

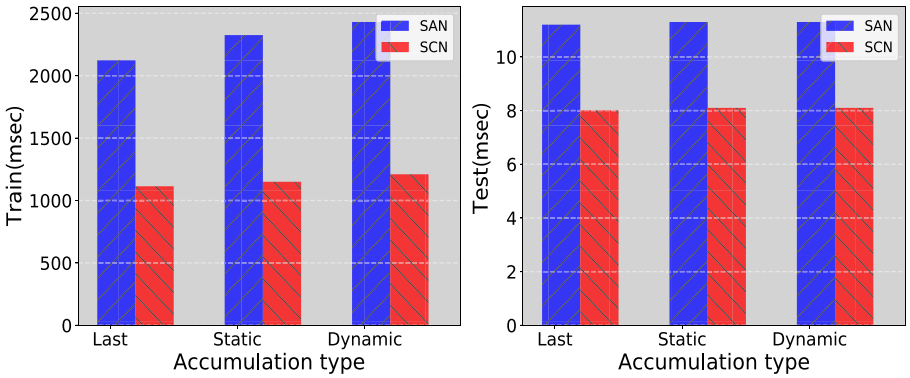


Figure 7
Efficiency of SCN and SAN. The left panel shows the training time per batch with 200 dimensional word embeddings, and the right panel shows the inference time per batch. One batch contains 200 instances.

corpus. This is because utterances in the Douban corpus are longer than those in the Ubuntu corpus (average length 18.5 vs. 12.4), which means single utterances in the Douban corpus could contain more information than those in the Ubuntu corpus. In practice, we set the maximum context length to 10 to balance effectiveness and efficiency.

7.6.4 Model Efficiency. In Section 5.1.3, we theoretically analyzed the efficiency of SCN and SAN. To verify the theoretical results, we further empirically compared their efficiency using the training data and the test data of the two data sets. The experiments were conducted using Theano on a Tesla K80 GPU with a Windows Server 2012 operation system. The parameters of the two models are described in Section 7.4. Figure 7 gives the training time and the test time of SAN and SCN. We can see that SCN is twice as fast as SAN in the training process (as a result of low time complexity and ease of parallelization), and saves 3 msec per batch in the test process. Moreover, different matching functions do not influence the running time as much, because the bottleneck is the utterance representation learning.

The empirical results are consistent with our theoretical results: SCN is faster than SAN. The results indicate that SCN is suitable for systems that care more about efficiency, whereas SAN can reach a higher accuracy with a little sacrifice of efficiency.

7.6.5 Visualization. We finally explained how SAN and SCN understand the semantics of conversation contexts by visualizing the similarity matrices of SCN, the attention weights of SAN, and the update gate and the reset gate of the accumulation GRU of the two models using an example from the Ubuntu corpus. Table 8 shows an example that is selected from the test set of the Ubuntu corpus and ranked at the top position by both SAN and SCN.

Figure 8(a) illustrates word–word similarity matrices $\mathbf{M}_1$ in SCN. We can see that important words in u_1 such as "unzip," "rar," and "files" are recognized and highlighted by words like "command," "extract," and "directory" in r. On the other hand, the similarity matrix of r and u_3 is almost blank, as there is no important information conveyed by u_3. Figure 8(b) shows the sequence-to-sequence similarity matrices $\mathbf{M}_2$ in SCN. We find that important segments like "unzip many rar" are highlighted, and the matrices

Table 8
An example for visualization from the Ubuntu corpus.

Context
u_1: how can unzip many rar files at once?
u_2: sure you can do that in bash
u_3: okay how?
u_4: are the files all in the same directory?
u_5: yes they all are;
Response
Response: then the command glebihan should extract them all from/to that directory

also provide complementary matching information to $\mathbf{M}_1$. Figure 8(c) visualizes the reset gate and the update gate of the accumulation GRU, respectively. Higher values in the update gate represent more information from the corresponding matching vector flowing into matching accumulation. From Figure 8(c), we can see that u_1 is crucial to response selection and nearly all information from u_1 and r flows to the hidden state of GRU, whereas other utterances are less informative and the corresponding gates are almost "closed" to keep the information from u_1 and r until the final state.

Regarding SAN, Figure 9(a) and Figure 9(b) illustrate the word level attention weights $\mathbf{A_1}$ and segment level attention weights $\mathbf{A_2}$, respectively. Similar to SCN, important words such as "zip" and "file" and important segments like "unzip many rar" get high weights, whereas function words like "that" and "for" are less attended. It should be noted that as the attention weights are normalized, the gaps between high and low values in $\mathbf{A_1}$ and $\mathbf{A_2}$ are not so large as those in $\mathbf{M}_1$ and $\mathbf{M}_2$ of SCN. Figure 9(c) visualizes the gates of the accumulation GRU, from which we observed similar distributions as those of SCN.

7.7 Error Analysis and Future Work

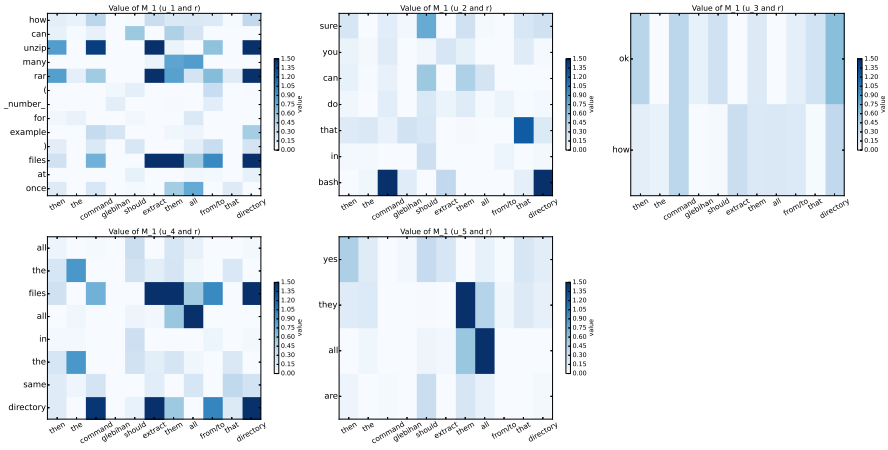
Although models under SMF outperform baseline methods on the two data sets, there are still several problems that cannot yet be handled perfectly.

(1) Logical consistency. SMF models the context and response on a semantic level, but pays little attention to logical consistency. This leads to several bad cases in the Douban corpus. We give a typical example in Table 9. In the conversation history, one of the speakers says that he thinks the item on Taobao is fake, and the response is expected to be why he dislikes the fake shoes. However, both SCN and SAN rank the response "It is not a fake. I just worry about the date of manufacture." at the top position. The response is inconsistent with the context in terms of logic, as it claims that the jogging shoes are not fake, which is contradictory to the context.

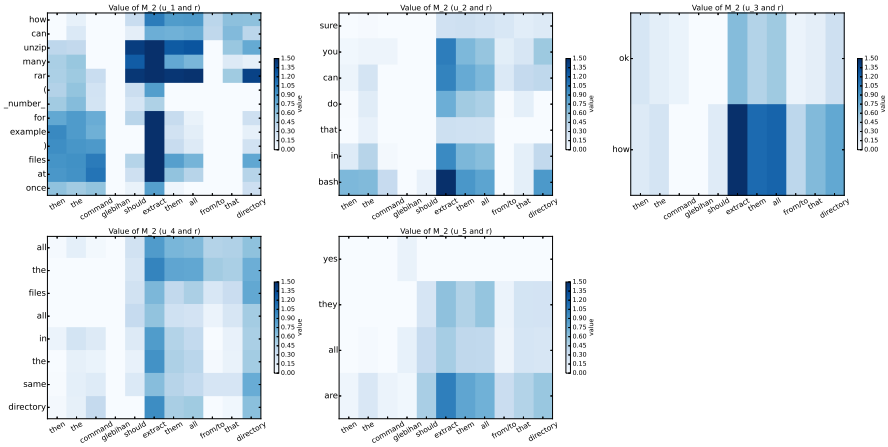
The reason behind this is that SMF only models semantics of context-response pairs. Logic, attitude, and sentiment are not taken into account in response selection.

In the future, we shall explore the logic consistency problem in retrieval-based chatbots by leveraging more features.

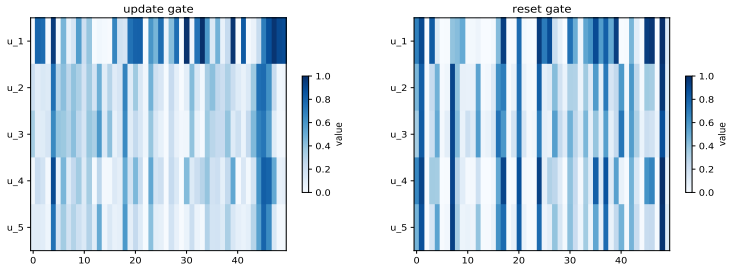
(2) No valid candidates. Another serious issue is the quality of candidates after retrieval. According to Wu et al. (2017), the candidate retrieval method can be described as follows: given a message u_n with $\{u_1, \ldots, u_{n-1}\}$ utterances in its previous turns, the



(a) Visualization of M_1 in SCN. Darker squares refer to higher values.

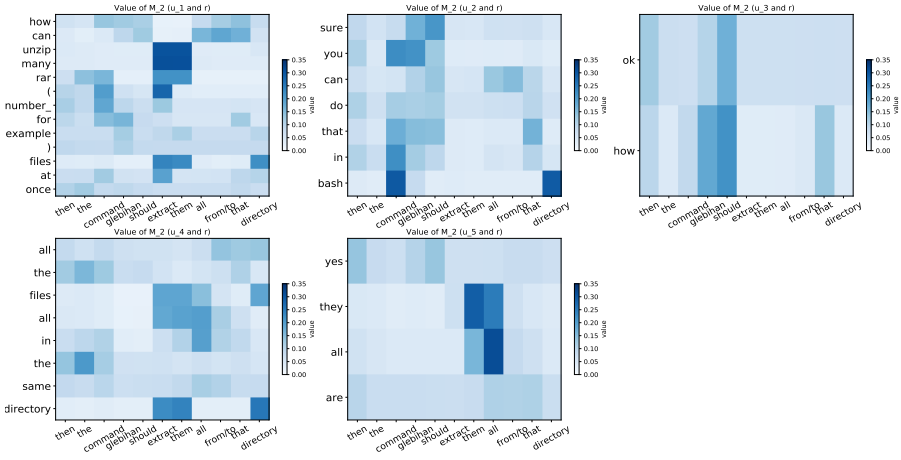


(b) Visualization of M_2 in SCN. Darker sqaures refer to higher values.

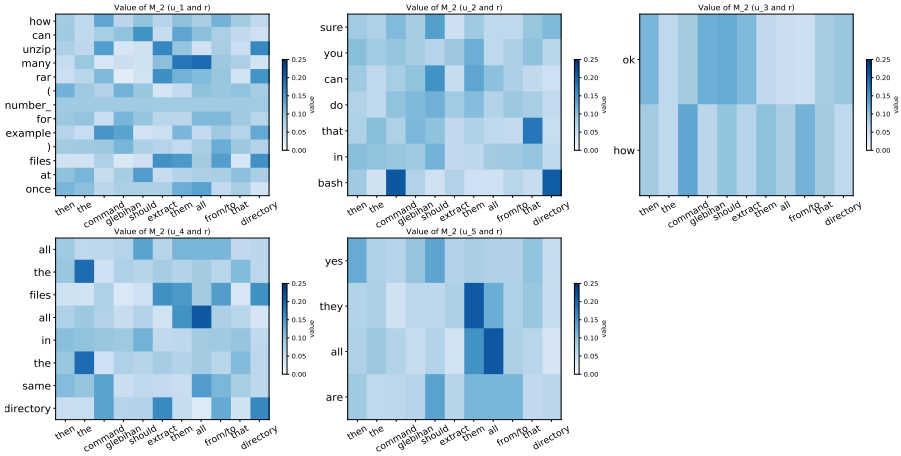


(c) Visualization of gates. Darker squares refer to higher values.

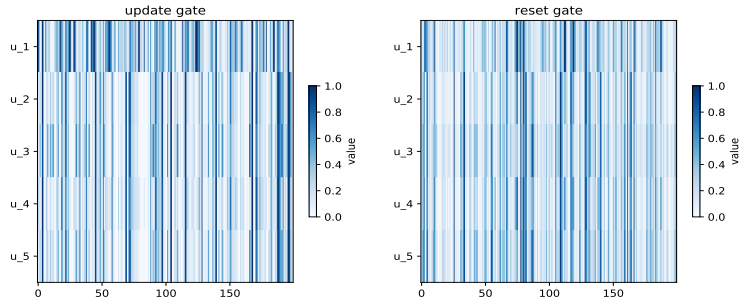
Figure 8
Visualization of SCN.



(a) Visualization of A_1 in SAN. Darker squares refer to higher values.



(b) Visualization of A_2 in SAN. Darker squares refer to higher values.



(c) Visualization of gates. Darker squares refer to higher values.

Figure 9
Visualization of SAN.

Table 9
An example in the Douban corpus. The response is ranked at the top position among candidates, but it is inconsistent on logic to the current context.

Context
u_1: Does anyone know Newton jogging shoes?
u_2: 100 RMB on Taobao.
u_3: I know that. I do not want to buy it because that is a fake which is made in Qingdao,
u_4:Is it the only reason you do not want to buy it?
Response
Response: It is not a fake. I just worry about the date of manufacture.

top five keywords are extracted from $\{u_1, \ldots, u_{n-1}\}$ based on their TF-IDF scores.[5] u_n is then expanded with the keywords, and the expanded message is sent to the index to retrieve response candidates using the inline retrieval algorithm of the index. The performance of the heuristic message expansion method is not good enough. In the experiment, only 667 out of 1,000 contexts have correct candidates after response candidate retrieval. This indicates that there is still much room to improve the retrieval component, and message expansion with several keywords from previous turns may not be enough for candidate retrieval. In the future, we will consider advanced methods for retrieving candidates.

(3) Gap between training and test. The current method requires a huge amount of training data (i.e., context-response pairs) to learn a matching model. However, it is too expensive to obtain large-scale (e.g., millions of) human labeled pairs in practice. Therefore, we regard conversations with human replies as positive instances and conversations with randomly sampled replies as negative instances in model training. The negative sampling method, however, oversimplifies the learning of a matching model because most negative candidates are semantically far from human responses, and thus easy to recognize; and some negative candidates might be proper responses if they are judged by a human. Because of the gap in training and test, our matching models, although performing much better than the baseline models, are still far from perfect on the Douban corpus (see the low P@1 in Table 5). In the future, we may consider using small human labeled data sets but leveraging the large-scale unlabeled data to learn matching models.

8. Conclusion

In this paper we studied the problem of multi-turn response selection in which one has to model the relationships among utterances in a context and pay more attention to important parts of the context. We find that the existing models cannot address the two challenges at the same time when we summarize them into a general framework.

5 Tf is word frequency in the context, and IDF is calculated using the entire index.

Motivated by the analysis, we propose a sequential matching framework for context-response matching. The new framework is able to capture the important information in a context and model the utterance relationships simultaneously. Under the framework, we propose two specific models based on a convolution-pooling technique and an attention mechanism. We test the two models on two public data sets. The results indicate that both models can significantly outperform the state-of-the-art models. To further understand the models, we conduct ablation analysis and visualize key components of the two models. We also compare the two models in terms of their efficacy, efficiency, and sensitivity to hyper-parameters.

Acknowledgments

Yu Wu is supported by an AdeptMind Scholarship and a Microsoft Scholarship. This work was supported in part by the Natural Science Foundation of China (grants U1636211, 61672081, 61370126), the Beijing Advanced Innovation Center for Imaging Technology (grant BAICIT-2016001), and the National Key R&D Program of China (grant 2016QY04W0802).

References

Baeza-Yates, Ricardo, Berthier Ribeiro-Neto, et al. 1999. *Modern Information Retrieval*, 463. ACM Press, New York.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Chen, Qian, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, and Hui Jiang. 2016. Enhancing and combining sequential and tree LSTM for natural language inference. *CoRR*, abs/1609.06038.

Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, Doha.

Chung, Junyoung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Elman, Jeffrey L. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Fleiss, Joseph L. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649, Vancouver.

He, Hua and Jimmy J. Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948, San Diego, CA.

Higashinaka, Ryuichiro, 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*, pages 928–939, Dublin.

Hochreiter, Sepp and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hu, Baotian, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050, Montreal.

Huang, Po-Sen, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 2333–2338, San Francisco, CA.

Ji, Zongcheng, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *CoRR*, abs/1408.6988.

Kadlec, Rudolf, Martin Schmid, and Jan Kleindienst. 2015. Improved deep learning baselines for Ubuntu corpus dialogs. *CoRR*, abs/1510.03753.

Kim, Yoon. 2014. Convolutional neural networks for sentence classification. In

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, pages 1746–1751, Doha.

Kingma, Diederik P. and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Li, Jiwei, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. 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*, pages 110–119, San Diego, CA.

Li, Jiwei, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, and William B. Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, pages 994–1003, Berlin.

Li, Jiwei, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016c. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*, pages 1192–1202, Austin, TX.

Li, Jiwei, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*, pages 2157–2169, Copenhagen.

Liu, Pengfei, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016a. Deep fusion LSTMs for text semantic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics Volume 1: Long Papers*, pages 1034–1043, Berlin.

Liu, Pengfei, Xipeng Qiu, Yaqian Zhou, Jifan Chen, and Xuanjing Huang. 2016b. Modelling interaction of sentence pair with coupled-LSTMs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*, pages 1703–1712, Austin, TX.

Lowe, Ryan, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. 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*, pages 285–294, Prague.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, Lake Tahoe, NV.

Mou, Lili, 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 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers*, pages 3349–3358, Osaka.

Parikh, Ankur P., Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*, pages 2249–2255, Austin, TX.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pages 1532–1543, Doha.

Qiu, Xipeng and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1305–1311, Buenos Aires.

Ritter, Alan, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 583–593, Edinburgh.

Rocktäschel, Tim, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664.

Serban, Iulian Vlad, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C. Courville. 2017a. Multiresolution recurrent neural networks: An application to dialogue response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3288–3294, San Francisco, CA.

Serban, Iulian Vlad, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models.

In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3776–3784, Phoenix, AZ.
Serban, Iulian Vlad, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301, San Francisco, CA.
Severyn, Aliaksei and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382, Santiago.
Shang, Lifeng, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL 2015, Volume 1: Long Papers*, pages 1577–1586, Beijing.
Shen, Yelong, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110, Shanghai.
Socher, Richard, Danqi Chen, Christopher D. Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934, Lake Tahoe, NV.
Sordoni, Alessandro, 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 HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205, Denver, CO.
Tan, Ming, Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Volume 1: Long Papers*, pages 464–473, Berlin.
Theano Development Team. 2016. Theano: A python framework for fast computation of mathematical expressions. *CoRR*, abs/1605.02688.
Vinyals, Oriol and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.
Voorhees, Ellen M. and Dawn M. Tice. 2000. The TREC-8 question answering track. In *Proceedings of the Second International Conference on Language Resources and Evaluation, LREC 2000*, pages 26–34, Athens.
Wan, Shengxian, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016a. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2835–2841, Phoenix, AZ.
Wan, Shengxian, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016b. Match-SRNN: Modeling the recursive matching structure with spatial RNN. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*, pages 2922–2928, New York, NY.
Wang, Bingning, Kang Liu, and Jun Zhao. 2016. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Volume 1: Long Papers*, pages 1288–1297, Berlin.
Wang, Hao, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*, pages 935–945, Seattle, WA.
Wang, Mingxuan, Zhengdong Lu, Hang Li, and Qun Liu. 2015. Syntax-based deep matching of short texts. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 1354–1361, Buenos Aires.
Wang, Shuohang and Jing Jiang. 2016a. A compare-aggregate model for matching text sequences. *CoRR*, abs/1611.01747.
Wang, Shuohang and Jing Jiang. 2016b. Learning natural language inference with LSTM. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1442–1451, San Diego, CA.
Weizenbaum, Joseph. 1966. ELIZA: a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.
Wu, Yu, Wei Wu, Zhoujun Li, and Ming Zhou. 2018a. Learning matching models with weak supervision for response

selection in retrieval-based chatbots. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Volume 2: Short Papers*, pages 420–425, Melbourne.

Wu, Yu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Volume 1: Long Papers*, pages 496–505, Vancouver.

Wu, Yu, Wei Wu, Dejian Yang, Can Xu, and Zhoujun Li. 2018b. Neural response generation with dynamic vocabularies. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 5594–5601, New Orleans, LA.

Xing, Chen, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3351–3357, San Francisco, CA.

Xing, Chen, Yu Wu, Wei Wu, Yalou Huang, and Ming Zhou. 2018. Hierarchical recurrent attention network for response generation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 5610–5617, New Orleans, LA.

Xu, Zhen, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2017. Incorporating loose-structured knowledge into conversation modeling via recall-gate LSTM. In *2017 International Joint Conference on Neural Networks, IJCNN 2017*, pages 3506–3513, Anchorage, AK.

Yan, Rui, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2016*, pages 55–64, Pisa.

Yin, Wenpeng and Hinrich Schütze. 2015. MultigranCNN: An architecture for general matching of text chunks on multiple levels of granularity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 63–73, Beijing.

Yin, Wenpeng, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *TACL*, 4:259–272.

Young, Steve, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174.

Zhou, Hao, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 730–739, New Orleans, LA.

Zhou, Xiangyang, 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 *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*, pages 372–381, Austin, TX.