Multi-Representation Fusion Network for Multi-turn Response Selection in Retrieval-based Chatbots

Chongyang Tao
ICST, Peking University
chongyangtao@pku.edu.com

Wei Wu
Microsoft Corporation
wuwei@microsoft.com

Can Xu
Microsoft Corporation
xu.can@microsoft.com

Wenpeng Hu
ICST, Peking University
wenpeng.hu@pku.edu.com

Dongyan Zhao
ICST, Peking University
Beijing Institute of Big Data Research
zhaody@pku.edu.com

Rui Yan[*]
ICST, Peking University
Beijing Institute of Big Data Research
ruiyan@pku.edu.com

ABSTRACT

We consider context-response matching with multiple types of representations for multi-turn response selection in retrieval-based chatbots. The representations encode semantics of contexts and responses on words, n-grams, and sub-sequences of utterances, and capture both short-term and long-term dependencies among words. With such a number of representations in hand, we study how to fuse them in a deep neural architecture for matching and how each of them contributes to matching. To this end, we propose a multi-representation fusion network where the representations can be fused into matching at an early stage, at an intermediate stage, or at the last stage. We empirically compare different representations and fusing strategies on two benchmark data sets. Evaluation results indicate that late fusion is always better than early fusion, and by fusing the representations at the last stage, our model significantly outperforms the existing methods, and achieves new state-of-the-art performance on both data sets. Through a thorough ablation study, we demonstrate the effect of each representation to matching, which sheds light on how to select them in practical systems.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**;

KEYWORDS

Fusing multiple representations; deep neural network; matching; multi-turn response selection; retrieval-based chatbot

ACM Reference Format:
Chongyang Tao, Wei Wu, Can Xu, Wenpeng Hu, Dongyan Zhao, and Rui Yan. 2019. Multi-Representation Fusion Network for Multi-turn Response Selection in Retrieval-based Chatbots. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), February 11–15, 2019, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3289600.3290985

Table 1: An example from Ubuntu Dialogue Corpus.

Turn-1	**Speaker A**:	hi how to install compiz fusion?
Turn-2	**Speaker B**:	i should be installed per default go to path, and enable the advanced desktop effects on the visual effects tab.
Turn-3	**Speaker A**:	tnks but i know there was a compiz fusion with lot more options, such as fire etc. where do i get it?
Turn-4	**Speaker B**:	open gome control enter and click on advanced desktop settings.
Turn-5	**Speaker A**:	sorry bothering you but where is gnome control center?
Turn-6	**Speaker B**:	enter it as a command into a terminal.
Response	**Speaker A**:	i have tried it. gnome control center is open but i cant find advanced desktop settings.

1 INTRODUCTION

Recent years have witnessed the prosperity of building dialogue systems for human-machine conversation. One type of such systems are task-oriented [45], which target on helping users accomplish specific tasks in vertical domains, such as train routing [3] and museum guiding [4], etc. Different from task-oriented dialogue systems, non-task-oriented chatbots aim to naturally and meaningfully converse with humans on any open domain topics [20]. With the rapid growth of human-human conversational data on social media, building a non-task-oriented chatbot with data-driven approaches is drawing more and more attention from both academia and industry. Existing methods are either retrieval-based or generation-based. The former retrieve a number of response candidates from a prebuilt index, and then select an appropriate one as a reply to a human input [27, 28, 33, 37, 38, 44], and the latter directly synthesize a response via natural language generation techniques [24, 25, 30]. In this work, we study the problem of response selection for retrieval-based chatbots, not only because retrieval-based methods have the advantage of providing fluent and informative responses, but also because they have been widely applied in many real products such as the social bot XiaoIce [26] from Microsoft and the E-commerce assistant AliMe Assist from Alibaba Group [9].

We consider multi-turn response selection, which requires a system to properly measure the matching degree between a conversational context and a response candidate. A conversational context consists of the current input and several turns before it, and thus renders complicated semantics. Therefore, the task is challenging in several aspects: first, the semantics of a context and a response is built upon language units from multiple granularities (e.g., words, phrases, and sub-sentences, etc.), and on each granularity, different units are differentially important for recognizing the semantic relationship between the context and the response; second, language in conversation is often informal, and sometimes contains grammatical errors and typos. Thus, units with a head meaning (e.g., "sorry") could be expressed in a tail form (e.g., "srry"); third, dependencies

[*]Corresponding author: Rui Yan (ruiyan@pku.edu.cn)

WSDM '19, February 11–15, 2019, Melbourne, VIC, Australia

ACM ISBN 978-1-4503-5940-5/19/02…$15.00
https://doi.org/10.1145/3289600.3290985

among language units exist in both short distance (e.g., a small window around a word) and long distance (e.g., words across the context and the response). Sometimes, meaning of a unit is clear only when it is related to its dependent ones, and distant but dependent units could indicate the semantic relationship between the context and the response. Table 1 illustrates the challenges with an example from the Ubuntu Dialogue Corpus [15] which is a benchmark data set used in our experiments. The conversation starts from "how" to install *compiz fusion*, and then moves to "where" to get *compiz fusion* with more options. Therefore, to accurately match the context with the response, one has to notice that words like "gnome" and "desktop" and phrases like "gnome control center" and "advanced desktop settings" are more important than words like "path" and "fire" and phrases like "install compiz fusion" and "visual effects tab" in the context. The inputs in the context are written in a very casual way with typos like "gome". In matching, it is better to connect "gome" with its correct expression "gnome" rather than to treat it as a new word. Moreover, "it" in the response refers to "command" in Turn-6 of the context. Such long-term dependency among words is helpful for establishing the relationship between the context and the response.

To tackle the challenges, we propose a multi-representation fusion network (MRFN) for context-response matching, where multiple types of representations of a context and a response are fused in a deep matching architecture. Each type of representation encodes semantics of units from a kind of granularity or a kind of dependency among the units. State-of-the-art methods, such as the sequential matching network (SMN) [38] and the deep attention matching network (DAM) [48], only exploit two types of representations, and thus there are no comprehensive studies on how a number of representations can be coordinated and leveraged in matching. Our MRFN follows a representation-matching-aggregation paradigm [38], and with many types of representations, we are particularly interested in two problems that are less explored before: (1) how to fuse the representations in matching ; and (2) how different types of representations contribute to the performance of matching.

Specifically, in the representation phase, we consider word representations, contextual representations, and attention-based representations. In word representations, each word is represented by the output of Word2Vec run on a specific corpus (e.g., the training set of the Ubuntu data) and a convolutional neural network (CNN) [7] on character embeddings. The former encodes co-occurrence information in the corpus, and the latter models morphology of words and could handle typos and informal expressions in conversation. In contextual representations, we represent utterances in a context and a response as sequences of vectors with a recurrent neural network with gated recurrent units (RNN-GRUs) [2] and a convolutional neural network on words. The GRU captures sequential relationship among words, and the CNN models local dependency among words (i.e., dependency within n-grams). Finally, in attention-based representations, we borrow the idea from Transformer [29] in machine translation, and represent utterances with self-attention and cross-attention. In self-attention, we let an utterance attend to itself in order to model dependency among words within the utterance. In cross-attention, we let the context and the response attend to each other. By this means, words that are important for matching are highlighted, and dependency of words across the context and the response is encoded in the representation.

When fusing different types of representations, we consider three strategies: fusing at an early stage (FES), fusing at an intermediate stage (FIS), and fusing at the last stage (FLS). Matching and aggregation are similar for different fusing strategies. In the matching phase, the model lets each utterance in the context interact with the response, and forms a matching vector for the utterance-response pair with a GRU operating on the representation of the interaction. Then in the aggregation phase, the matching vectors are processed by another GRU, and the last hidden state of the GRU is fed to a multi-layer perceptron (MLP) to calculate a matching score for the context and the response. The difference among the three strategies is that in FES, different types of representations are combined as one representation before the matching phase starts; in FIS, the interaction between the context and the response is performed upon each type of representations separately, and then the fusion of representations happens before the matching vectors are formed; and finally in FLS, both the matching phase and the aggregation phase are conducted on each type of representations, and matching scores from different representations are combined as a final score.

We conduct experiments on two benchmark data sets for multi-turn response selection: the Ubuntu Dialog Corpus [15] and the Douban Conversation Corpus [38]. Experimental results indicate that the later the representations are fused, the better the performance of matching will be. By fusing the proposed representations at the last stage, our MRFN significantly outperforms DAM [48], which is a strong baseline published very recently on ACL 2018, and thus achieves new state-of-the-art performance on both data sets. On the Ubuntu data, the improvement to DAM on $R_{10}@1$ (equivalent to P@1) is 1.9%, and on the Douban data, the improvement to DAM on P@1 is 2.1%. In terms of contributions of different representations, empirical results indicate that contextual representations and attention-based representations are more useful than word representations on both data sets. Our code is available at https://github.com/chongyangtao/MRFN.

To sum up, our contributions are four-folds: (1) proposal of fusing multiple types of representations for context-response matching with a multi-representation fusion network; (2) proposal of three strategies for representation fusion and empirical comparison of the strategies on benchmark data sets; (3) new state-of-the-art performance on benchmark data sets and publication of source code; and (4) thorough experiments that provide insights into contributions of different types of representations to matching.

2 RELATED WORK

Research of chatbots could date back to 1960s when ELIZA [36] was designed with handcrafted templates and heuristic rules. Recently, since large scale human-human conversational data becomes available on the Internet, researchers begin to study building a chatbot with data-driven approaches. Existing methods can be categorized into two groups. The first group of methods learn response generation models from the data. Early work employs the technique from statistical machine translation [20]. Recently, neural network based models have become the mainstream. On top of the sequence-to-sequence with attention architecture [24, 30], various extensions have been made to address the "safe response" problem [10]; to leverage external knowledge [18, 40]; to model the hierarchy of conversational contexts [22, 23, 41]; to generate responses with specific persona or emotions [11, 46]; to speed up response decoding [39]; and to pursue better optimization strategies [12, 13]. Different from the generation-based methods, the second group of methods select a proper response from the existing data through retrieval and ranking. A key step in these methods is measuring the matching degree between an input and a response candidate. Early work in this group studies single-turn response selection where the input is a single message [5, 6, 16, 33]. Recently more attention is drawn to context-response matching for multi-turn response selection. Representative methods include the dual LSTM model [15], the deep learning to respond architecture [43], the multi-view matching model [47], the sequential matching network (SMN) [38], and the deep attention matching network (DAM) [48].

In this work, we propose a multi-representation fusion network for multi-turn response selection. Our model belongs to the second group, and is unique in that contexts and responses are represented from six perspectives and all these representations are fused for matching. Some of the representations, such as the character-based embedding and CNN on words, are not well explored in previous models. In addition to pursuing state-of-the-art performance on benchmark data sets, we explore different fusing methods and empirically compare the effect of different representations in matching.

3 MODEL

3.1 Problem Formalization

Suppose that we have a data set $\mathcal{D} = \{c_i, r_i, y_i\}_{i=1}^N$, where $c_i = \{u_{i,1}, u_{i,2}, \ldots, u_{i,m_i}\}$ represents a conversational context with $u_{i,k}$ the k-th utterance; r_i denotes a response candidate; and $y_i \in \{0, 1\}$ is a label. $y_i = 1$ means that r_i is a proper response for c_i, otherwise, $y_i = 0$. Our goal is to learn a matching model $g(\cdot, \cdot)$ from $\mathcal{D}$, and thus for any new context $c = \{u_1, \ldots, u_m\}$ (m is the number of utterances in the context) and a response candidate r, $g(c, r)$ measures the matching degree between c and r.

Given an utterance u, which is either a $u_i \in c$ or the response candidate r, let $u = (w_1, \ldots, w_n)$ where w_j is the j-th word, we assume that u can be represented as $\{\boldsymbol{U}^k\}_{k=1}^K$. $\forall k$, $\boldsymbol{U}^k = (\boldsymbol{e}_1^k, \ldots, \boldsymbol{e}_n^k)$ is a type of representation of u with $\boldsymbol{e}_j^k$ a vector corresponding to w_j. We aim to define $g(\cdot, \cdot)$ with $\{\boldsymbol{U}^k\}_{k=1}^K$. To this end, we need to address two problems: (1) how to define $\{\boldsymbol{U}^k\}_{k=1}^K$ which can encode various semantics and dependencies in c and r; and (2) how to leverage $\{\boldsymbol{U}^k\}_{k=1}^K$ to perform matching. In the following sections, we will first describe the definition of representations, and then elaborate the matching architectures built upon the representations.

3.2 Representations

We consider three groups of representations: word representations, contextual representations, and attention-based representations. These representations encode semantic information in words, n-grams, and sub-sentences, and capture both short-term dependency and long-term dependency among words.

3.2.1 Word Representations.

Word representations encode semantics of individual words. Here, we aim to capture similarity of words in terms of both semantics and morphology. Hence, we consider the following two types of representations:

Character-based Word Embedding. Given a word w, suppose that $w = (x_1, \ldots, x_L)$, where x_i is the i-th character of w, we first obtain the embedding of x_i by looking up a table, and form an embedding matrix for w. Then, we apply a 1-dimensional convolutional neural network (CNN) [7] to the embedding matrix, and calculate the embedding of w by alternating convolution and max pooling operations. Let d_e denote the number of filters of CNN, then for the j-th filter with a window size s_j, the output of the convolution operation is $(o_{j,1}, \ldots, o_{j,L})$[1] with the t-th element as

$$o_{j,t} = tanh\Big(\boldsymbol{W}_c^j \cdot \boldsymbol{x}_{t:t+s_j-1} + \boldsymbol{b}_c^j\Big), \tag{1}$$

where $\boldsymbol{x}_{t:t+s_j-1}$ refers to the concatenation of the embeddings of $(x_t, \ldots, x_{t+s_j-1})$, and $\boldsymbol{W}_c^j$ and $\boldsymbol{b}_c^j$ are parameters. A max pooling operation follows the convolution operation and is formulated as

$$\hat{o}_j = \max\{o_{j,1}, \ldots, o_{j,L}\}. \tag{2}$$

w is then represented as $(\hat{o}_1, \ldots, \hat{o}_{d_e})$. In practice, character embedding is randomly initialized and updated with other parameters.

CNN models composition of characters in a window, and encodes sub-word information into its output. Thus, different forms of a word, such as "gome" and "gnome" in Table 1, can be connected via their embeddings, and the out-of-vocabulary issue can be mitigated.

Word2Vec. In addition to the character-based embedding, we also represent words by running Word2Vec [17] on a specific corpus (e.g., the training set of a benchmark data set). The representation encodes co-occurrence information in the corpus, and lays the foundation for other representations in the following. In learning, word vectors (dimension=d_w) are initialized with Word2Vec and then updated with other parameters. Note that we do not concatenate the character-based embedding with Word2Vec as previous studies [21], since we aim to investigate effect of the representations individually[2]. One can also replace Word2Vec with similar algorithms such as GloVe [19]. We leave the study as future work.

3.2.2 Contextual Representations.

Contextual representations encode semantics of n-grams and sub-sequences in utterances. N-grams represent a kind of local context of a word, and sub-sequences represent a kind of sequential context of a word. Thus, we denote the former as "local representation" and the latter as "sequential representation". Both representations focus on capturing short-term dependency among words.

Sequential Representation. We employ a recurrent neural network with gated recurrent units (RNN-GRUs) to transform $u = (w_1, \ldots, w_n)$ into a sequence of hidden vectors $(\boldsymbol{h}_1, \cdots, \boldsymbol{h}_n)$. $\forall t \in \{1, \ldots, n\}$, $\boldsymbol{h}_t \in \mathbb{R}^{d_s}$ is formulated as

$$\begin{aligned} [\boldsymbol{r}_t; \boldsymbol{z}_t] &= \sigma(\boldsymbol{W}_{r,z}\boldsymbol{e}(w_t) + \boldsymbol{V}_{r,z}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{r,z}), \\ \tilde{\boldsymbol{h}}_t &= \tanh\Big(\boldsymbol{W}_h\boldsymbol{e}(w_t) + \boldsymbol{V}_h(\boldsymbol{r}_t \circ \boldsymbol{h}_{t-1}) + \boldsymbol{b}_h\Big), \\ \boldsymbol{h}_t &= (1 - \boldsymbol{z}_t) \circ \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \circ \tilde{\boldsymbol{h}}_t, \end{aligned} \tag{3}$$

[1] We pad zeros to keep the dimension.

[2] We tried the method in [21], and the performance of our model drops drastically.

where $\boldsymbol{e}(w_t)$ is the Word2Vec embedding of w_t, $\boldsymbol{r}_t$ and $\boldsymbol{z}_t$ are a reset gate and an update gate respectively, $\sigma(\cdot)$ is a sigmoid function, $\circ$ is element-wise multiplication, and $\boldsymbol{W}_{r,z,h}$, $\boldsymbol{V}_{r,z,h}$, $\boldsymbol{b}_{r,z,h}$ are parameters. $\boldsymbol{h}_t$ encodes sequential relationship and dependency among words up to position t in utterance u. $\boldsymbol{h}_0$ is randomly initialized.

Local Representation. We transform $u = (w_1, \ldots, w_n)$ to a sequence of vectors $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n)$ with a CNN on $(\boldsymbol{e}(w_1), \ldots, \boldsymbol{e}(w_n))$. By varying window size of the filters in $\{1, 2, 3, 4\}$, CNN models unigrams, bigrams, trigrams, and four-grams respectively. For each size, there are n_c filters whose parameters are not shared and initialized independently, and these filters output an n_c-dimensional vector ($n_c = 50$ in our experiments) for each position of u through a convolution operation similar to Equation (1). The difference is that character embeddings are now replaced by word embeddings. $\forall t \in \{1, \ldots, n\}$, $\boldsymbol{c}_t \in \mathbb{R}^{d_l}$ is finally defined by a concatenation of outputs from all filters on position t ($d_l = 4 * n_c$). $\boldsymbol{c}_t$ encodes semantics and dependency of words in n-grams (e.g., "gnome control center" in Table 1).

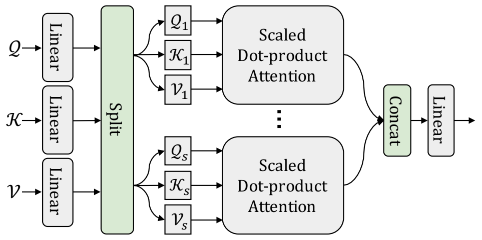


Figure 1: Multi-head attention.

3.2.3 *Attention-based Representations.*

Attention-based representations encode importance of words and long-term dependency among words with an attention-mechanism.

Multi-head Attention. Before diving to details of the representations, we first introduce multi-head attention [14, 29] which is the basis of all attention-based representations. Figure 1 gives the architecture of multi-head attention, in which the key component is a scaled dot-product attention module. Formally, let $\mathcal{Q} \in \mathbb{R}^{n_{\mathcal{Q}} \times d}$, $\mathcal{K} \in \mathbb{R}^{n_{\mathcal{K}} \times d}$, and $\mathcal{V} \in \mathbb{R}^{n_{\mathcal{V}} \times d}$ denote embedding matrices of a query, a key, and a value respectively, where $n_{\mathcal{Q}}$, $n_{\mathcal{K}}$, and $n_{\mathcal{V}}$ refer to the numbers of words in the three inputs respectively, d is the dimension of the embedding, and $n_{\mathcal{K}} = n_{\mathcal{V}}$. In this work, we initialize the embedding matrices with Word2Vec and update the matrices with other parameters. The scaled dot-product attention module is then defined by

$$\text{Attention}(\mathcal{Q}, \mathcal{K}, \mathcal{V}) = \text{softmax}\left(\frac{\mathcal{Q}\mathcal{K}^T}{\sqrt{d}}\right)\mathcal{V} \tag{4}$$

Intuitively, in Equation (4), each entry of $\mathcal{V}$ is weighted by an importance score defined by the similarity of an entry of $\mathcal{Q}$ and an entry of $\mathcal{K}$. The entries of $\mathcal{V}$ are then linearly combined with the weights to form a new representation of $\mathcal{Q}$. In practice, we usually let $\mathcal{K} = \mathcal{V}$. Thus, a word in $\mathcal{Q}$ is represented by its most similar words in $\mathcal{V}$. $\mathcal{Q}$, $\mathcal{K}$, and $\mathcal{V}$ are dispensed to s heads. Each head models relationship among $\mathcal{Q}$, $\mathcal{K}$, and $\mathcal{V}$ from one aspect, and corresponds to a scaled dot-product attention module. $\forall 1 \leqslant i \leqslant s$, the output of head i is given by

$$\mathcal{Z}_i = \text{Attention}(\mathcal{Q}\boldsymbol{W}_i^{\mathcal{Q}}, \mathcal{K}\boldsymbol{W}_i^{\mathcal{K}}, \mathcal{V}\boldsymbol{W}_i^{\mathcal{V}}), \tag{5}$$

where $\boldsymbol{W}_i^{\mathcal{Q}}, \boldsymbol{W}_i^{\mathcal{K}}, \boldsymbol{W}_i^{\mathcal{V}} \in \mathbb{R}^{d \times (d/s)}$ are linear transformations. Then the output of multi-head attention is defined by

$$O = (\mathcal{Z}_1 \oplus \mathcal{Z}_2 \oplus \cdots \oplus \mathcal{Z}_s)\boldsymbol{W}_o \tag{6}$$

where $\oplus$ means row-wise concatenation of two matrices, and $\boldsymbol{W}_o \in \mathbb{R}^{d \times d}$ is a linear transformation.

In our model, we also add a residual connection to O (i.e., $O = O + \mathcal{Q}$) and carry out row-wise normalization (with a mean of 0 and a standard deviation of 1). For ease of presentation, we denote multi-head attention module as MultiHeadAttention$(\mathcal{Q}, \mathcal{K}, \mathcal{V})$ and consider two types of representations by varying $\mathcal{Q}$, $\mathcal{K}$, and $\mathcal{V}$. Note that representations obtained by this module have the same dimension with the query $\mathcal{Q}$.

Self-attention. We set $\mathcal{Q}, \mathcal{K}$, and $\mathcal{V}$ as $\mathcal{U} = (\boldsymbol{e}(w_1), \ldots, \boldsymbol{e}(w_n))$. Then, utterance u (either from the context or the response candidate) is represented as MultiHeadAttention$(\mathcal{U}, \mathcal{U}, \mathcal{U})^\top$. Self-attention lets an utterance attend to itself, and represents each word by other similar words within the utterance. As a result, the representation could encode dependency among distant words in the utterance, such as "fire""compiz", and "it" in Turn-3 of the context in Table 1.

Cross-attention. Given an utterance u_i in context c and response candidate r, let $\mathcal{U}_i = (\boldsymbol{e}(w_{i,1}), \ldots, \boldsymbol{e}(w_{i,j}), \ldots, \boldsymbol{e}(w_{i,n_i}))$ and $\mathcal{R} = (\boldsymbol{e}(w_{r,1}), \ldots, \boldsymbol{e}(w_{r,j}), \ldots, \boldsymbol{e}(w_{r,n_r}))$ be embedding matrices of u_i and r respectively, where $w_{i,j}$ and $w_{r,j}$ are the j-th words of u_i and r respectively, and $\boldsymbol{e}(w_{i,j})$ and $\boldsymbol{e}(w_{r,j})$ are their embeddings obtained from Word2Vec. We then define the representation of u_i as MultiHeadAttention$(\mathcal{U}_i, \mathcal{R}, \mathcal{R})^\top$, and the representation of r as MultiHeadAttention$(\mathcal{R}, \mathcal{U}_i, \mathcal{U}_i)^\top$.

In cross-attention, words in the context (response) are represented by similar words in the response (context). Thus, dependent words across the context and the response could be connected via the representation (e.g., in Table 1, "command" in Turn-6 of the context, and "it" in the response). Moreover, since the representation is learned from training data, words in the context that are useful for matching will have large similarity with words in the response, and thus play an important role in the response representation.

3.3 Multi-Representation Fusion Network

For any utterance u_i in a context c, we denote $\{\boldsymbol{U}_i^k\}_{k=1}^K$ as K ($K = 6$) types of representations of u_i, where $\{\boldsymbol{U}_i^k\}_{k=1}^K$ are defined in Section 3.2. Similarly, we denote $\{\boldsymbol{R}^k\}_{k=1}^K$ as the K types of representations of a response candidate r. Then, the matching model $g(\cdot,\cdot)$ is defined in a deep neural architecture with $\{\boldsymbol{U}_i^k\}_{k=1}^K$, $1 \leqslant i \leqslant m$ and $\{\boldsymbol{R}^k\}_{k=1}^K$ as inputs. In a nutshell, $g(\cdot,\cdot)$ works in the following way: each $u_i \in c$ interacts with r at the beginning, and the pair of (u_i, r) is transformed to an interaction representation $\boldsymbol{T}_i$. Then, $\boldsymbol{T}_i$ is processed by a GRU as a matching vector $\boldsymbol{v}_i$. Finally, all matching vectors $\{\boldsymbol{v}_i\}_{i=1}^m$ are aggregated as a matching score $g(c, r)$. Here, since there are multiple types of representations, in addition to matching and aggregation, we also have to investigate how to fuse the representations in order to sufficiently leverage each of them in matching. To this end, we consider three fusing strategies, namely fusing at an early stage (FES), fusing at an intermediate stage (FIS), and fusing at the last stage (FLS). Figure 2 illustrates the architecture of our model with the three fusing strategies. The model features

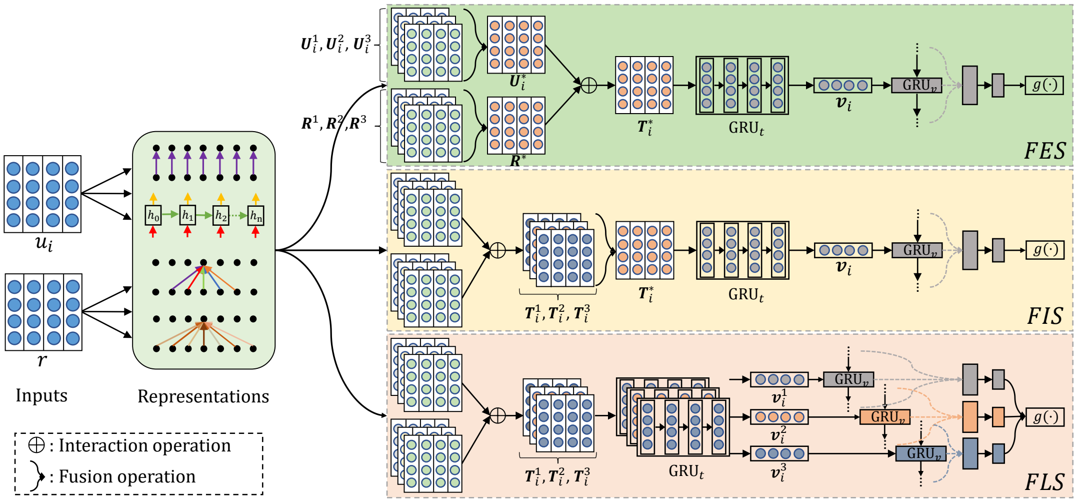


Figure 2: Multi-representation fusion network with three fusing strategies (*FES, FIS, FLS*). u_i is the i-th utterance in a context and r is a response candidate. We only draw three representations for each utterance for ease of illustration in this figure.

by a deep structure that fuses multiple types of representations for matching, therefore, we name it multi-representation fusion network (MRFN).

3.3.1 Fusing at an early stage (FES). In FES, the model fuses all representations before interaction of utterance-response pairs starts. Formally, $\forall u_i \in c$, $\{\boldsymbol{U}_i^k\}_{k=1}^K$ is collapsed into a single representation $\boldsymbol{U}_i^*$. $\forall k \in \{1, \ldots, K\}$, suppose that $\boldsymbol{U}_i^k$ is a $(d_k \times n_i)$-dimensional matrix with $\boldsymbol{e}_{i,j}^k \in \mathbb{R}^{d_k}$ the j-th column corresponding to the j-th word of u_i and n_i the number of words of u_i, then the j-th column of $\boldsymbol{U}_i^* \in \mathbb{R}^{d^* \times n_i}$ $(d^* = \sum_k d_k)$ is defined by $\oplus_{k=1}^K \boldsymbol{e}_{i,j}^k$, where $\oplus$ is a concatenation operator. Similarly, r is represented as $\boldsymbol{R}^* \in \mathbb{R}^{d^* \times n_r}$ whose j-th column is given by $\oplus_{k=1}^K \boldsymbol{e}_{r,j}^k$, where $\boldsymbol{e}_{r,j}^k \in \mathbb{R}^{d_k}$ is the j-th column of $\boldsymbol{R}^k$ and n_r is the number of words of r.

The interaction of u_i and r is performed with $\boldsymbol{U}_i^*$ and $\boldsymbol{R}^*$, and can be decomposed into two steps: we first let $\boldsymbol{U}_i^*$ attend to $\boldsymbol{R}^*$ and form an intermediate representation for u_i; then we create an interaction representation for (u_i, r) by an interaction function with $\boldsymbol{U}_i^*$ and the intermediate representation as two arguments. Specifically, suppose that $\hat{\boldsymbol{e}}_{i,j}$ and $\hat{\boldsymbol{e}}_{r,k}$ are the j-th and k-th columns of $\boldsymbol{U}_i^*$ and $\boldsymbol{R}^*$ respectively, then the attention weight of $\hat{\boldsymbol{e}}_{r,k}$ regarding to $\hat{\boldsymbol{e}}_{i,j}$ is defined by

$$\omega_{j,k}^i = \boldsymbol{V}_a^\top \tanh(\boldsymbol{W}_a[\hat{\boldsymbol{e}}_{i,j} \oplus \hat{\boldsymbol{e}}_{r,k}] + \boldsymbol{b}_a)$$

$$\alpha_{j,k}^i = \frac{\exp\,(\omega_{j,k}^i)}{\sum_{k=1}^{n_r} \exp\,(\omega_{j,k}^i)}, \tag{7}$$

where $\boldsymbol{W}_a, \boldsymbol{V}_a$, and $\boldsymbol{b}_a$ are parameters. Let us denote $\overline{\boldsymbol{e}}_{i,j} \in \mathbb{R}^{d^*}$ as the j-th column of the intermediate representation for u_i $(j \in \{1, \ldots, n_i\})$, then $\overline{\boldsymbol{e}}_{i,j}$ is formulated as

$$\overline{\boldsymbol{e}}_{i,j} = \textstyle\sum_{k=1}^{n_r} \alpha_{j,k}^i \hat{\boldsymbol{e}}_{r,k}. \tag{8}$$

Finally, $\boldsymbol{U}_i^*$ and the intermediate representation are fed to an interaction function to form $\boldsymbol{T}_i$. Here, we employ the *SUBMULT+NN* function proposed in [35] as the interaction function, as the function has proven effective in various tasks. Suppose that $\boldsymbol{T}_i = (\boldsymbol{t}_{i,1}, \ldots, \boldsymbol{t}_{i,n_i}) \in \mathbb{R}^{d^* \times n_i}$, then, the j-th column of $\boldsymbol{T}_i$ is defined as

$$\boldsymbol{t}_{i,j} = f(\hat{\boldsymbol{e}}_{i,j}, \overline{\boldsymbol{e}}_{i,j}) = \mathrm{ReLU}(\boldsymbol{W}_p \begin{bmatrix} (\hat{\boldsymbol{e}}_{i,j} - \overline{\boldsymbol{e}}_{i,j}) \odot (\hat{\boldsymbol{e}}_{i,j} - \overline{\boldsymbol{e}}_{i,j}) \\ \hat{\boldsymbol{e}}_{i,j} \odot \overline{\boldsymbol{e}}_{i,j} \end{bmatrix} + \boldsymbol{b}_p), \tag{9}$$

where $\odot$ refers to element-wise multiplication, and $\boldsymbol{W}_p$ and $\boldsymbol{b}_p$ are parameters. Note that the interaction of u_i and r is realized through an attention mechanism given by Equation (7) and Equation (8), which is similar to cross-attention. In our experiments, we find that the effect of cross-attention is not fully covered by the interaction of utterance-response pairs. The sequence of $(\boldsymbol{t}_{i,1}, \ldots, \boldsymbol{t}_{i,n_i})$ is then boiled down to a matching vector $\boldsymbol{v}_i \in \mathbb{R}^{d_v}$ through a GRU. Suppose that the sequence of hidden states of the GRU is $(\boldsymbol{h}_1^t, \ldots, \boldsymbol{h}_{n_i}^t)$, then $\forall j, 1 \leqslant j \leqslant n_i, \boldsymbol{h}_j^t \in \mathbb{R}^{d_v}$ is defined by

$$\boldsymbol{h}_j^t = \mathrm{GRU}_t(\boldsymbol{h}_{j-1}^t, \boldsymbol{t}_{i,j}), \tag{10}$$

where the parameterization of $\mathrm{GRU}_t(\cdot,\cdot)$ is similar to Equation (3). $\boldsymbol{v}_i$ is defined as $\boldsymbol{h}_{n_i}^t$. $\boldsymbol{h}_0^t$ is randomly initialized in our experiments.

After we have $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ as the matching vectors of $((u_1, r)$ $\ldots, (u_m, r))$, we employ another GRU to model the temporal relationship of the utterances in context c. Suppose that the hidden states of the GRU are $(\boldsymbol{h}_1^v, \ldots, \boldsymbol{h}_m^v)$, then $\forall j, 1 \leqslant j \leqslant m, \boldsymbol{h}_j^v \in \mathbb{R}^{d_v}$ is given by

$$\boldsymbol{h}_j^v = \mathrm{GRU}_v(\boldsymbol{h}_{j-1}^v, \boldsymbol{v}_j), \tag{11}$$

where $\mathrm{GRU}_v(\cdot,\cdot)$ has a similar parameterization with Equation (3) and $\boldsymbol{h}_0^v$ is randomly initialized. The matching score of (c, r) is defined by a multi-layer perceptron (MLP) with $\boldsymbol{h}_m^v$ as an input:

$$g(c, r) = f_2(\boldsymbol{W}_2^\top \cdot f_1(\boldsymbol{W}_1^\top \boldsymbol{h}_m^v + \boldsymbol{b}_1) + \boldsymbol{b}_2), \tag{12}$$

where $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{b}_1$, and $\boldsymbol{b}_2$ are parameters. $f_1(\cdot)$ is the tanh activation function, and $f_2(\cdot)$ is the softmax function.

3.3.2 Fusion at an intermediate stage (FIS). In FIS, utterance-response interaction is executed on each type of representations, which results in K interaction representations. Then, the K representations are fused to one representation, which is used to calculate a matching vector for the utterance-response pair. Specifically, let

$\boldsymbol{T}_i^k$ be the k-th interaction representation of (u_i, r), which is calculated by Equation (7-9) with $\boldsymbol{U}_i^k$ and $\boldsymbol{R}^k$ as inputs. $\forall k \in \{1, \ldots, K\}$, suppose that $\boldsymbol{T}_i^k = (\boldsymbol{t}_{i,1}^k, \ldots, \boldsymbol{t}_{i,n_i}^k) \in \mathbb{R}^{d_k \times n_i}$, then $\boldsymbol{T}_i^k$ is collapsed to a $\boldsymbol{T}_i^* = (\boldsymbol{t}_{i,1}^*, \ldots, \boldsymbol{t}_{i,n_i}^*) \in \mathbb{R}^{d^* \times n_i}$ ($d^* = \sum_k d_k$) with the j-th column $\boldsymbol{t}_{i,j}^* \in \mathbb{R}^{d^*}$ as $\oplus_{k=1}^K \boldsymbol{t}_{i,j}^k$. The remaining procedure of FIS is the same with that of FES: $\boldsymbol{T}_i^*$ is transformed to a matching vector $\boldsymbol{v}_i$ by $\text{GRU}_t(\cdot, \cdot)$, and then $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ are aggregated by $\text{GRU}_v(\cdot, \cdot)$ and an MLP to form $g(c, r)$.

3.3.3 *Fusion at the last stage (FLS).*

FLS decomposes the matching of (c, r) to K pipelines, where each pipeline takes one type of representations as inputs and outputs a matching score for (c, r). The K matching scores are finally fused as $g(c, r)$. Specifically, for each (u_i, r), $\boldsymbol{U}_i^k$ and $\boldsymbol{R}^k$ are first transformed to $\boldsymbol{T}_i^k \in \mathbb{R}^{d_k \times n_i}$ by Equation (7-9). $\boldsymbol{T}_i^k$ is then fed to $\text{GRU}_t(\cdot, \cdot)$ whose last hidden state is taken as a matching vector $\boldsymbol{v}_i^k \in \mathbb{R}^{d_k}$. Finally, $(\boldsymbol{v}_1^k, \ldots, \boldsymbol{v}_m^k)$ are aggregated as a $g^k(c, r)$ via $\text{GRU}_v(\cdot, \cdot)$ and an MLP. With $\{g^k(c, r)\}_{k=1}^K$, we define $g(c, r)$ as

$$g(c, r) = \textstyle\sum_{k=1}^K g^k(c, r). \tag{13}$$

3.4 Learning

We learn $g(\cdot, \cdot)$ by minimizing cross entropy with $\mathcal{D}$. Let Θ denotes the parameters, then the learning objective of FES and FIS is:

$$\mathcal{J}_\Theta = -\textstyle\sum_{i=1}^N y_i \log(g(c_i, r_i)) + (1 - y_i) \log(1 - g(c_i, r_i)). \tag{14}$$

For FLS, the learning objective is a summation of learning objectives of K pipelines:

$$\mathcal{J}_\Theta = -\sum_{k=1}^K \sum_{i=1}^N y_i \log(g^k(c_i, r_i)) + (1 - y_i) \log(1 - g^k(c_i, r_i)). \tag{15}$$

All the learning objectives are optimized using back-propagation with Adam algorithm [8].

4 EXPERIMENTS

4.1 Experiment Setup

We test MRFN on two benchmark data sets. The first one we use is the Ubuntu Dialogue Corpus [15] (Ubuntu v1.0) which consists of English multi-turn conversations about technical support collected from chat logs of the Ubuntu forum. We use the copy shared in [42], in which numbers, paths and URLs are replaced by placeholders. The training set contains 1 million context-response pairs with the ratio of positive examples and negative examples as $1:1$. Both the validation set and the test set contain 0.5 million pairs with the ratio of positive examples and negative examples as $1:9$. In all the three sets, a positive example consists of a context and a response coming from a human, and a negative example is constructed by randomly sampling a response from the entire data set for a context. Following [15, 38, 48], we employ $R_n@k$s as evaluation metrics, where $R_n@k$ stands for recall at position k in n candidates, and measures if the positive response can be ranked in top k positions when there are n candidates.

In addition to the Ubuntu data, we use the Douban Conversation Corpus [38] as another data set for experiments. The data contains open domain multi-turn conversations in Chinese crawled from Douban group[3], which is a popular social networking service in China. In the data set, there are 1 million context-response pairs for training, $50,000$ pairs for validation, and $6,670$ pairs for test. In the training data and the validation data, the last turn of each conversation is regarded as a positive response, and negative responses are randomly sampled from the entire corpus. For each context, the ratio of positive examples and negative examples is $1:1$. In the test data, each context corresponds to 10 responses retrieved from an index. Different from the Ubuntu data, the appropriateness of a response regarding to a context in the test data is labeled by human annotators, and one context could have multiple positive response candidates. Following [38], we employ mean average precision (MAP) [1], mean reciprocal rank (MRR) [31], precision at position 1 (P@1), and $R_n@k$s as evaluation metrics.

4.2 Baseline Models

The following models are selected as baselines for comparison:

Basic deep matching models [32, 34]: these models, including MV-LSTM [32], and Match-LSTM [34], perform context-response matching by concatenating all utterances in a context as a long document and calculating a matching score with the long document and a response candidate.

Multi-View [47]: the model measures the matching degree between a context and a response candidate in both a word view and an utterance view.

DL2R [43]: the model reformulates a message with previous turns in a context with multiple approaches. A response candidate and the reformulated message are then represented by a composition of an RNN and a CNN. Finally, the matching score is computed with the concatenation of the representations.

SMN [38]: the model lets each utterance in a context interacts with a response candidate at the first step, and then transforms each utterance-response pair into a matching vector with CNNs. The matching vectors are finally aggregated with an RNN as a matching score of the context and the response candidate.

DAM [48]: the state-of-the-art model until this submission. The architecture of the model is similar to that of SMN, but utterances in the context and the response are represented with stacked self-attention and cross attention.

In comparison, we copy the numbers from [38, 48] for the baseline models. In order to conduct statistical tests and discussion, we implement SMN and DAM following the settings reported in [38] and [48] respectively. Our implementation achieves comparable performance with the models reported in the two papers.

4.3 Implementation Details

In our model, we set the number of filters as 100 (i.e., d_e) on both data when computing the char-based word embedding with CNN. On the Ubuntu data, the window size of each filter is 5, and on the Douban data, the window size is 3. Word2Vec are run on the training sets of the two data with the dimension of the word embedding as 200 (i.e., d_w). In multi-head attention, we set the number of heads as 8 (i.e., s) on both data, following [29]. The hidden size of GRU in sequential representation (i.e., d_s) is 200. We tune the hidden sizes of GRU_t and GRU_v in FES and FIS (i.e., d_v) in

[3] https://www.douban.com/group

Table 2: Evaluation results on the two data sets. Numbers in bold mean that the improvement is statistically significant compared with the best baseline.

	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
DL2R [43]	0.899	0.626	0.783	0.944	0.488	0.527	0.330	0.193	0.342	0.705
MV-LSTM [32]	0.906	0.653	0.804	0.946	0.498	0.538	0.348	0.202	0.351	0.710
Match-LSTM [34]	0.904	0.653	0.799	0.944	0.500	0.537	0.345	0.202	0.348	0.720
Multi-View [47]	0.908	0.662	0.801	0.951	0.505	0.543	0.342	0.202	0.350	0.729
SMN [38]	0.926	0.726	0.847	0.961	0.529	0.569	0.397	0.233	0.396	0.724
DAM [48]	0.938	0.767	0.874	0.969	0.550	0.601	0.427	0.254	0.410	0.757
MRFN_{FES}	0.930	0.742	0.857	0.963	0.538	0.583	0.405	0.244	0.410	0.740
MRFN_{FIS}	0.936	0.762	0.870	0.967	0.558	0.605	0.438	0.267	0.417	0.759
MRFN_{FLS}	**0.945**	**0.786**	**0.886**	**0.976**	**0.571**	0.617	**0.448**	**0.276**	**0.435**	**0.783**

{400, 600, 800, 1000, 1200}, and find that 1000 is the best choice. The hidden sizes of GRU_t and GRU_v in FLS are kept the same as the dimension of the corresponding representation. In both data, the size of the vocabulary is the same as previous studies [38, 48]. Similar to [38], we limit the length of contexts to 10 turns and the length of utterances to 50 words. If the number of turns in a context is less than 10, we pad zeros, otherwise, the latest 10 turns are kept. The maximum number of characters of a word is set as 16 in the Ubuntu data and 8 in the Douban data. In optimization, we choose 100 as the size of mini-batches, and $1e-3$ the learning rate. Early stopping on the validation data is adopted as a regularization strategy.

4.4 Evaluation Results

Comparison of fusing strategies. Table 2 reports the evaluation results of all models on both data sets. First, MRFN_{FES}, MRFN_{FIS}, and MRFN_{FLS} show a consistent trend on both data sets over all metrics (i.e., $\text{MRFN}_{\text{FES}} < \text{MRFN}_{\text{FIS}} < \text{MRFN}_{\text{FLS}}$). From the results, we can conclude that the later the representations are fused, the better the performance of the matching model will be. Our explanation to the phenomenon is that there is information loss when a representation passes through the components of the deep architecture of MRFN. Thus, the earlier different types of representations are fused together, the more information of contexts and responses will be lost in matching. Second, MRFN_{FLS} outperforms the best baseline DAM in terms of all metrics on both data sets. We conduct statistical tests, and the results indicate that the improvement on all metrics except MRR on the Douban data is statistically significant (t-test with p-value < 0.05). Third, MRFN_{FIS} is worse than DAM on the Ubuntu data, but it is better than DAM on the Douban data, which might stem from the small test set of the Douban data.

Impact of different representations. We conduct a comprehensive ablation study to investigate the impact of different representations. Table 3 reports the results. First, we remove each representation individually from MRFN_{FLS}, and denote the models as MRFN_{FLS}-X, where X ∈ {Character, Word2Vec, Seq, Conv, Self, Cross} meaning character-based word embedding, Word2Vec, sequential representation, local representation, self-attention, and cross-attention respectively. We find that each representation is useful, but removing one only leads to slight performance drop. The reason might be that the information conveyed by the representations is redundant. For example, the effect of Conv might be covered by that of Seq, because of the reset and update mechanism of GRU. In spite of this, we can still conclude that on the Ubuntu data, the rank of the representations in terms of R_{10}@1 is that: Self > Seq = Character > Conv > Cross = Word2Vec; and on the Douban data, the rank of the representations in terms of P@1 is that: Self = Seq > Cross > Conv > Word2Vec > Character. Character is more important on the Ubuntu data than it is on the Douban data, because English has morphology, and conversations of the Ubuntu data contain more typos and informal expressions than conversations of the Douban data. We select R_{10}@1 and P@1 as target metrics in the study of importance of representations, because they are more critical than other metrics in real systems of response selection. Note that we do not remove the representations one by one, because different orders by which the representations are removed may result in different conclusions on importance of the representations.

Second, we remove each group of representations individually, and denote the models as MRFN_{FLS}-Word, MRFN_{FLS}-Contextual, and MRFN_{FLS}-Attention, meaning that word representations, contextual representations, and attention-based representations are removed respectively. By this means, we can alleviate the influence of information redundancy in the study. We find that contextual representations and attention-based representations are more useful than word representations, because removing either of them results in significant performance drop on R_{10}@1 and P@1 (t-test with p-value < 0.05). The conclusion can be further verified when we only keep one group of representations in matching (the models are denoted as MRFN_{FLS} with Word, MRFN_{FLS} with Contextual, and MRFN_{FLS} with Attention), since MRFN_{FLS} with Contextual and MRFN_{FLS} with Attention are better than MRFN_{FLS} with Word in terms of all metrics (on the Ubuntu data, the improvement is significant over all metrics under t-test with p-value < 0.05). MRFN_{FLS} with Attention achieves comparable performance with DAM in terms of most metrics on both data, but our model is more efficient than DAM in both training and test: on average, MRFN_{FLS} with Attention is 1.9x faster than DAM in training and 1.7x faster than DAM in test. This is because we do not have to stack many attentive modules like DAM.

4.5 Discussion

Does the conclusion regarding to fusing strategies hold on other architectures? We are curious if the conclusion of "early fusion < intermediate fusion < late fusion" depends on the architecture of our model. Therefore, we test the three fusing strategies

Table 3: Evaluation results of representation ablation on the two data sets.

	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
MRFN$_{\text{FLS}}$	0.945	0.786	0.886	0.976	0.571	0.617	0.448	0.276	0.435	0.783
MRFN$_{\text{FLS}}$-Character	0.943	0.783	0.885	0.971	0.566	0.611	0.448	0.273	0.425	0.770
MRFN$_{\text{FLS}}$-Word2vec	0.944	0.785	0.887	0.972	0.562	0.611	0.445	0.266	0.419	0.780
MRFN$_{\text{FLS}}$-Seq	0.943	0.783	0.884	0.970	0.563	0.607	0.433	0.261	0.425	0.776
MRFN$_{\text{FLS}}$-Conv	0.944	0.784	0.885	0.973	0.564	0.610	0.444	0.267	0.421	0.777
MRFN$_{\text{FLS}}$-Self	0.943	0.782	0.883	0.972	0.560	0.605	0.433	0.263	0.420	0.777
MRFN$_{\text{FLS}}$-Cross	0.944	0.785	0.885	0.972	0.571	0.609	0.439	0.266	0.411	0.782
MRFN$_{\text{FLS}}$-Word	0.943	0.781	0.883	0.971	0.559	0.603	0.433	0.262	0.417	0.768
MRFN$_{\text{FLS}}$-Contextual	0.940	0.777	0.881	0.970	0.554	0.599	0.427	0.255	0.411	0.768
MRFN$_{\text{FLS}}$-Attention	0.941	0.778	0.881	0.970	0.558	0.603	0.426	0.254	0.423	0.776
MRFN$_{\text{FLS}}$ with Word	0.929	0.746	0.856	0.960	0.537	0.585	0.403	0.236	0.402	0.753
MRFN$_{\text{FLS}}$ with Contextual	0.938	0.771	0.875	0.967	0.551	0.594	0.417	0.250	0.421	0.768
MRFN$_{\text{FLS}}$ with Attention	0.937	0.767	0.874	0.968	0.549	0.593	0.412	0.249	0.419	0.759

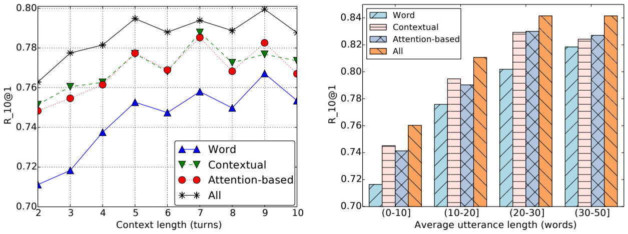
Table 4: Performance comparison over different fusing strategies for SMN [38].

	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
SMN$_{\text{FES}}$	0.915	0.692	0.831	0.956	0.518	0.566	0.381	0.225	0.377	0.727
SMN$_{\text{FIS}}$	0.926	0.726	0.847	0.961	0.529	0.569	0.397	0.233	0.396	0.724
SMN$_{\text{FLS}}$	0.932	0.744	0.863	0.965	0.545	0.586	0.406	0.250	0.406	0.751

on the SMN model [38]. The original SMN model proposed in [38] fuses Word2Vec and the sequential representation at an intermediate stage. Thus, we implement SMN models with early-stage fusion and last-stage fusion, where for the former, we concatenate Word2Vec and the sequential representation before the formation of the similarity matrix, and for the latter, we calculate a matching score with each type of representations and combine them as we do in Equation (13). Table 4 reports the comparison results on both data sets. We observe a consistent trend with MRFN on SMN. The results indicate that our conclusion on fusing strategies generally holds on various matching architectures.

Performance across different context length. We further study how the number of turns and the length of utterances influence the performance of MRFN$_{\text{FLS}}$ and its variants (MRFN$_{\text{FLS}}$ with Word, MRFN$_{\text{FLS}}$ with Contextual, and MRFN$_{\text{FLS}}$ with Attention). Figure 3(a) shows how the performance of the models changes with respect to different average numbers of turns of contexts. We observe a similar trend for all models: they first increase monotonically until context length reaches 5, and then fluctuate when context length keeps increasing. The reason might be that when only a few utterances are available in contexts, the model could not capture enough information for matching, but when the contexts become long enough, noise will be brought to matching as utterances in early history could be irrelevant to the current input. MRFN$_{\text{FLS}}$ with Contextual and MRFN$_{\text{FLS}}$ with Attention are comparable over different context lengths and both of them are better than MRFN$_{\text{FLS}}$ with Word, which is consistent with the result in Table 3.

Figure 3(b) illustrates how the performance of the models changes with respect to contexts with different average length of utterances. We find that: (1) MRFN$_{\text{FLS}}$ with Word is the worst over all intervals; (2) contextual representations are more useful than attention-based representations when the average length of utterances is less than 20. However, the conclusion is reversed when the average length of utterances increases. This is because that attention-based representations are better at capturing long-term dependency among words, while contextual representations are better at modeling short-term or local dependency among words; (3) utilizing all representations can always improve the performance of matching.



(a) Average turns Vs. R10@1 (b) Average utterance length Vs. R10@1

Figure 3: Performance of MRFN$_{\text{FLS}}$ and its variants across different length of contexts. (a) context length is measured by average number of turns; and (b) context length is measured by average length of utterances.

5 CONCLUSION AND FUTURE WORK

We perform context-response matching with multiple types of representations in a multi-representation fusion network. The representations encode semantics of words, n-grams, and sub-sequences, and capture both short-term and long-term dependencies among words. In the model, the representations can be fused into matching at an early stage, at an intermediate stage, and at the last stage. Empirical studies on two benchmark data sets indicate that late fusion is better than early fusion, and by fusing the representations at the last stage, our model achieves new state-of-the-art performance on both data sets. In addition to that, we also investigate the effect of different representations through extensive ablation studies. In the future, we would like to explore more complicated representations, such as deep contextual representations and attention-based representations learnt by stacked architectures.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their constructive comments. This work was supported by the National Key Research and Development Program of China (No. 2017YFC0804001), the National Science Foundation of China (NSFC Nos. 61672058 and 61876196). Rui Yan was sponsored by Microsoft Research Asia (MSRA) Collaborative Research Program.

REFERENCES

[1] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*. Vol. 463. ACM press New York.

[2] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555* (2014).

[3] George Ferguson, James F Allen, Bradford W Miller, et al. 1996. TRAINS-95: Towards a Mixed-Initiative Planning Assistant.. In *AIPS*. 70–77.

[4] Nadine Glas, Ken Prepin, and Catherine Pelachaud. 2015. Engagement Driven Topic Selection for An Information-giving Ggent. In *Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2015-goDial)*.

[5] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *Advances in Neural Information Processing Systems*. 2042–2050.

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

[7] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1746–1751.

[8] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for Stochastic Optimization. In *ICLR*.

[9] Feng-Lin Li, Minghui Qiu, Haiqing Chen, Xiongwei Wang, Xing Gao, Jun Huang, Juwei Ren, Zhongzhou Zhao, Weipeng Zhao, Lei Wang, et al. 2017. AliMe Assist: An Intelligent Assistant for Creating an Innovative E-commerce Experience. In *Proceedings of the Conference on Information and Knowledge Management*. 2495–2498.

[10] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A Diversity-Promoting Objective Function for Neural Conversation Models. In *NAACL*. 110–119.

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

[12] Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep Reinforcement Learning for Dialogue Generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 1192–1202.

[13] Jiwei Li, Will Monroe, Tianlin Shi, Sèbastien 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*. 2157–2169.

[14] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-attentive Sentence Embedding. In *ICLR*.

[15] Ryan Lowe, 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 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 285–294.

[16] Zhengdong Lu and Hang Li. 2013. A Deep Architecture for Matching Short Texts. In *Advances in Neural Information Processing Systems*. 1367–1375.

[17] Tomas Mikolov, 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*. 3111–3119.

[18] Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to Backward and Forward Sequences: A Content-Introducing Approach to Generative Short-Text Conversation. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*. 3349–3358.

[19] Jeffrey Pennington, 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*. 1532–1543.

[20] Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven Response Generation in Social Media. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. 583–593.

[21] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional Attention Flow for Machine Comprehension. In *ICLR*.

[22] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. In *AAAI*. 3776–3784.

[23] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues. In *AAAI*. 3295–3301.

[24] Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural Responding Machine for Short-Text Conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. 1577–1586.

[25] Mingyue Shang, Zhenxin Fu, Nanyun Peng, Yansong Feng, Dongyan Zhao, and Rui Yan. 2018. Learning to Converse with Noisy Data: Generation with Calibration.. In *IJCAI*. 4338–4344.

[26] Heung-Yeung Shum, Xiaodong He, and Di Li. 2018. From Eliza to XiaoIce: Challenges and Opportunities with Social Chatbots. *Frontiers of IT & EE* 19, 1 (2018), 10–26.

[27] Chongyang Tao, Shen Gao, Mingyue Shang, Wei Wu, Dongyan Zhao, and Rui Yan. 2018. Get The Point of My Utterance! Learning Towards Effective Responses with Multi-Head Attention Mechanism.. In *IJCAI*. 4418–4424.

[28] Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. RUBER: An Unsupervised Method for Automatic Evaluation of Open-Domain Dialog Systems. In *AAAI*. 722–729.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[30] Oriol Vinyals and Quoc Le. 2015. A Neural Conversational Model. *arXiv preprint arXiv:1506.05869* (2015).

[31] Ellen M Voorhees et al. 1999. The TREC-8 Question Answering Track Report.. In *Trec*, Vol. 99. 77–82.

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

[33] Hao Wang, 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*. 935–945.

[34] Shuohang Wang and Jing Jiang. 2016. Learning Natural Language Inference with LSTM. In *NAACL*. 1442–1451.

[35] Shuohang Wang and Jing Jiang. 2017. A Compare-Aggregate Model for Matching Text Sequences. In *ICLR*.

[36] Joseph Weizenbaum. 1966. ELIZA: A Computer Program for the Study of Natural Language Communication Between Man and Machine. *Commun. ACM* 9, 1, 36–45.

[37] Yu Wu, wei, Zhoujun Li, and Ming Zhou. 2018. 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*. 420–425.

[38] Yu Wu, 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*. 496–505.

[39] Yu Wu, Wei Wu, Dejian Yang, Can Xu, Zhoujun Li, and Ming Zhou. 2018. Neural Response Generation with Dynamic Vocabularies. In *AAAI*. 5594–5601.

[40] Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic Aware Neural Response Generation.. In *AAAI*. 3351–3357.

[41] Chen Xing, Wei Wu, Yu Wu, Ming Zhou, Yalou Huang, and Wei-Ying Ma. 2018. Hierarchical Recurrent Attention Network for Response Generation. In *AAAI*. 5610–5617.

[42] Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2017. Incorporating Loose-Structured Knowledge into LSTM with Recall Gate for Conversation Modeling. In *Proceedings of the 2017 International Joint Conference on Neural Networks*. 3506–3513.

[43] Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to Respond with Deep Neural Networks for Retrieval-Based Human-Computer Conversation System. In *SIGIR*. 55–64.

[44] Rui Yan and Dongyan Zhao. 2018. Coupled context modeling for deep chit-chat: towards conversations between human and computer. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2574–2583.

[45] Stephanie Young, Milica Gasic, Blaise Thomson, and John D Williams. 2013. POMDP-based Statistical Spoken Dialog Systems: A Review. *Proc. IEEE* 101, 5 (2013), 1160–1179.

[46] Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional Chatting Machine: Emotional Conversation Generation with Internal and External Memory. In *AAAI*. 730–738.

[47] Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, R Yan, D Yu, Xuan Liu, and H Tian. 2016. Multi-view Response Selection for Human-computer Conversation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 372–381.

[48] Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018. Multi-Turn Response Selection for Chatbots with Deep Attention Matching Network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. 1118–1127.