Knowledge Enhanced Hybrid Neural Network for Text Matching

Yu Wu[†◇], **Wei Wu**[‡], **Can Xu**[‡],**Zhoujun Li**[†◇*]
[†]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[‡] Microsoft Research, Beijing, China
[◇]Authors are supported by AdeptMind Scholarship
{wuyu,dejianyang,lizj}@buaa.edu.cn {wuwei,can.xu}@microsoft.com

Abstract

Long text brings a big challenge to neural network based text matching approaches due to their complicated structures. To tackle the challenge, we propose a knowledge enhanced hybrid neural network (KEHNN) that leverages prior knowledge to identify useful information and filter out noise in long text and performs matching from multiple perspectives. The model fuses prior knowledge into word representations by knowledge gates and establishes three matching channels with words, sequential structures of text given by Gated Recurrent Units (GRUs), and knowledge enhanced representations. The three channels are processed by a convolutional neural network to generate high level features for matching, and the features are synthesized as a matching score by a multilayer perceptron. In this paper, we focus on exploring the use of taxonomy knowledge for text matching. Evaluation results from extensive experiments on public data sets of question answering and conversation show that KEHNN can significantly outperform state-of-the-art matching models and particularly improve matching accuracy on pairs with long text.

Introduction

Text matching is a fundamental problem in many NLP tasks such as question answering (QA) (Voorhees and others 1999), retrieval based conversation (Wang et al. 2013), and paraphrase identification (Dolan, Quirk, and Brockett 2004).

The challenge of text matching lies in semantic gaps between text pairs. State-of-the-art methods tackle the challenge by representing text pairs or their semantic relations from different levels of abstractions with neural networks (Hu et al. 2014). The problem is that when text becomes long, which often happens in many applications, the performance of these methods drops dramatically. For example, in a public QA data set where we select a positive answer for a question according to their matching degree, the state-of-the-art model only achieves 70.6% matching accuracy on pairs longer than 30 words (Q+A) compared to its performance 78.8% on pairs shorter than 30 words. Similarly, in a public dialog corpus where we select a proper response for a message based on their matching degree, the state-of-the-art model achieves 38.7% R_{10}@1 (recall at position 1 in 10 candidates) on pairs longer than 20 words (M+R) compared to 41.1% R_{10}@1 on pairs shorter than 20 words. These numbers indicate that long text raises a new challenge for text matching and influence the neural network performance.

The reason state-of-the-art matching models perform badly on long text is that long text often has complicated structures which might hinder the models from accurately capturing semantic relations of the text pairs. For example, semantics of long text may reside in words, phrases, clauses and sentences. More seriously, much information in long text is often useless or even noise to matching. Table 1 shows an example from community question answering (CQA) to illustrate the challenge. The answer is very long[1] and contains a lot of information that well compare the two schools but semantically far from the question (e.g., horse riding and lances swords). The information makes the answer a high quality one, but interferes the existing models in question-answer matching.

We aim to improve matching accuracy on long text. Our idea is that we introduce prior knowledge into matching and perform matching from multiple perspectives including words, local structures of text such as phrases and clauses, and global context of text. The global context is represented by the prior knowledge which could be topics, tags, and entities related to the text pair and obtained elsewhere. In matching, the representation of text is constructed under the supervision of the prior knowledge, and thus important parts in text can be highlighted and noise can be filtered out. For instance, in CQA (e.g., Yahoo! Answers and Quora), askers are required to assign tags to their questions as a summarization of their semantics. The question in Table 1 was assigned a tag "Family" by the asker, and we can use it as prior knowledge to enhance the matching between the question and the answer. "Family" reflects the global semantics of the question. It can help strengthen the effect of its semantically similar words like "kids","child" and "activity" in QA matching, and at the same time reduce the influence of "horse riding" and "lances swords" to matching. With the tag as a bridge, the semantic relation between the question and the answer can be identified, which is difficult to achieve only by themselves.

[*]Corresponding Author.

[1]The original answer has 149 words.

Question : Which school is better Voltaire or Bonaparte?
Tag from the asker: Family
An appropriate answer : Both are good schools but Bonaparte will teach your kids to become a good leader but they concentrate mainly on outdoor physical activities, manoeuvers, strategies. Horse riding and lances swords are their speciality.... On the other hand Voltaire will make your child more of a philosopher! They encourage independent thinking...and mainly concentrates on indoor activities! They inculcate good moral values in the child and he will surely grow up to be a thinking person!

Table 1: An example from CQA

Unlike the existing work (Severyn and Moschitti 2015) where prior knowledge is used as an extra feature of neural networks and absent from matching until the last step, we propose a method lets prior knowledge operate on representations directly and affect matching from the beginning, namely knowledge enhanced hybrid neural network (KEHNN). KEHNN exploits a knowledge gate to fuse the semantic information carried by the prior knowledge into each word representation. The knowledge gate is a non-linear unit and controls how much information from the word is kept in the new representation and how much information from the prior knowledge flows to the representation. By this means, noise from the irrelevant words is filtered out, and useful information from the relevant words is highlighted. The model then forms three channels to perform matching from multiple perspectives. Each channel models the interaction of two pieces of text in a pair by a similarity matrix. The first channel matches text pairs on words. It calculates the similarity matrix by word embeddings. The second channel conducts matching on local structures of text. It captures sequential structures of text in the pair and models dependencies among words by a Bidirectional Recurrent Neural Network with Gated Recurrent Units (BiGRU) (Cho et al. 2014), and constructs the similarity matrix with the hidden vectors given by BiGRU. In the last channel, the knowledge enhanced representations, after processed by another BiGRU, are utilized to construct the similarity matrix. Since the prior knowledge represents global semantics of the text pair, the channel performs matching from a global context perspective. The three channels then exploit a convolutional neural network (CNN) to extract compositional relations of the matching elements as high level features for matching. The features are finally synthesized as a matching score by a multilayer perceptron (MLP).

We conduct experiments on public data sets of CQA and conversation. Evaluation results show that KEHNN can significantly outperform state-of-the-art matching methods, and particularly improve the matching accuracy on long text.

Our contributions in this paper are three-folds: 1) proposal of improving matching accuracy on long text; 2) proposal of a knowledge enhanced hybrid neural network which is a principled approach to leveraging prior knowledge and multiple levels of semantics of text in matching; 3) empirical verification of the effectiveness of the proposed method on two public data sets.

Related Work

Neural networks have proven effective on capturing semantic relations of text pairs. Existing work can be categorized into three groups. The first group follows a paradigm that text pairs are first individually represented by neural networks, and then the matching score is computed by the representations. Typical models in this group include DSSM (Huang et al. 2013), NTN (Socher et al. 2013), CDSSM (Shen et al. 2014), Arc1 (Hu et al. 2014), CNTN (Qiu and Huang 2015), and LSTM (Lowe et al. 2015) etc. The second group applies attention mechanisms to text matching. Representative models in this group include Match-LSTM (Wang and Jiang 2015), Inner-Attention (Wang, Liu, and Zhao 2016) and Attention LSTM (Tan, Xiang, and Zhou 2015; Rocktäschel et al. 2015) etc. The third group matches text pairs by their interaction matrices. For example, MV-LSTM (Wan et al. 2015) generates the matrix by LSTMs and neural tensors, and then uses k-max pooling and a multi-layer perceptron to compute a matching score. MultiGranCNN (Yin and Schütze 2015) constructs an interaction matrix with different length of text chunks, and employs a CNN to extract matching features. More effort along this line includes Unfolding RAE + Dynamic Pooling (Socher et al. 2011), ABCNN (Yin et al. 2015), Arc2 (Hu et al. 2014), Match-SRNN (Wan et al. 2016), CubeCNN (He and Lin 2016), DF-LSTM (Liu et al. 2016) and Coupled-LSTM (Liu, Qiu, and Huang 2016) etc. Our method falls into the third group. The major difference of KEHNN is that it leverages prior knowledge and multiple channels to enhance matching of long text. Before us, some researchers have proposed using prior knowledge as additional features in neural networks for text matching (Severyn and Moschitti 2015; Ghosh et al. 2016). Different from these work, our model leverages prior knowledge to improve representations of text, and thus noise in the text can be filtered out and important information can be highlighted in matching.

Matching Approach

Problem Formalization

Suppose that we have a $\mathcal{D} = \{(l_i, S_{x,i}, S_{y,i})\}_{i=1}^{N}$, where $S_{x,i} = (w_1, \ldots, w_j, \ldots, w_I)$ and $S_{y,i} = (w_1^{'}, \ldots, w_j^{'}, \ldots, w_J^{'})$ are two pieces of text, and w_j and $w_j^{'}$ represent the j-th word of $S_{x,i}$ and $S_{y,i}$ respectively. $l_i \in \{1, \ldots, C\}$ is a label indicating the matching degree between $S_{x,i}$ and $S_{y,i}$, where C is the scale of the matching degree. In addition to $\mathcal{D}$, we have prior knowledge for $S_{x,i}$ and $S_{y,i}$ denoted as $\mathbf{k}_{x,i}$ and $\mathbf{k}_{y,i}$ respectively. Our goal is to learn a matching model $g(\cdot,\cdot)$ with $\mathcal{D}$ and $\{\cup_{i=1}^{N}\mathbf{k}_{x,i}, \cup_{i=1}^{N}\mathbf{k}_{y,i}\}$. Given a new pair (S_x, S_y) with prior knowledge $(\mathbf{k}_x, \mathbf{k}_y)$, $g(S_x, S_y)$ predicts the matching degree between S_x and S_y.

In the following sections, we first assume the existence of prior knowledge and present our matching model. Then, we describe details on how to acquire prior knowledge in practice.

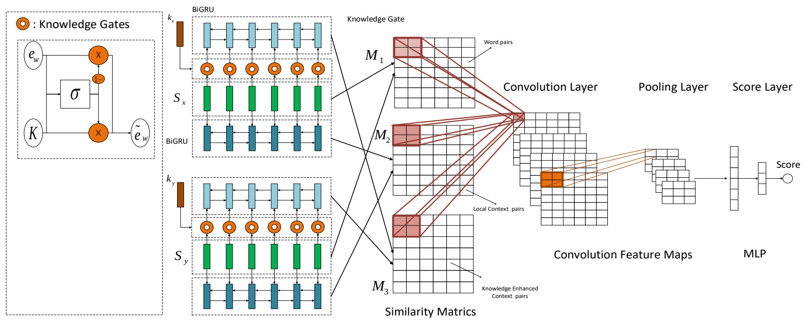


Figure 1: Architecture of KEHNN

Knowledge Gate

Inspired by the powerful gate mechanism (Hochreiter and Schmidhuber 1997; Chung et al. 2014) in recurrent neural networks (RNN), we propose using knowledge gates to incorporate prior knowledge into matching. The knowledge gate provides a general and principled approach to leverage prior knowledge in matching. The underlying motivation is that we want to use the prior knowledge to filter out noise and highlight the useful information to matching in a piece of text. Formally, let $e_w \in \mathbb{R}^d$ denote the embedding of a word w in text S_x and $\mathbf{k}_x \in \mathbb{R}^n$ denote the representation of the prior knowledge of S_x. Knowledge gate k_w is defined as

$$k_w = \sigma(W_k e_w + U_k \mathbf{k}_x), \tag{1}$$

where σ is a sigmoid function, and $W_k \in \mathbb{R}^{d \times d}$, $U_k \in \mathbb{R}^{d \times n}$ are parameters. With k_w, we define a knowledge enhanced representation for w as

$$\widetilde{e}_w = k_w \odot e_w + (1 - k_w) \odot \mathbf{k}_x, \tag{2}$$

where $\odot$ is an element-wise multiplication operation. Equation (2) means that prior knowledge is fused into matching by a combination of the word representation and the knowledge representation. In the combination, the knowledge gate element-wisely controls how much information from word w is preserved, and how much information from prior knowledge $\mathbf{k}_x$ flows in. The advantage of the element-wise operation is that it offers a way to precisely control the contributions of prior knowledge and words in matching. Entries of k_w lie in $[0, 1]$. The larger an entry of k_w is, the more information from the corresponding entry of e_w will be kept in $\widetilde{e}_w$. In contrast, the smaller an entry of k_w is, the more information from the corresponding entry of $\mathbf{k}_x$ will flow into $\widetilde{e}_w$. Since k_w is determined by both e_w and $\mathbf{k}_x$ and learned from training data, it will keep the useful parts in the representations of w and the prior knowledge and at the same time filter out noise from them.

Matching with Multiple Channels

With the knowledge enhanced representations, we propose a knowledge enhanced hybrid neural network (KEHNN) which conducts matching with multiple channels. Figure 1 gives the architecture of KEHNN. Given a pair (S_x, S_y), the model looks up an embedding table and represents S_x and S_y as $\mathbf{S}_x = [e_{x,1}, \ldots, e_{x,i}, \ldots, e_{x,I}]$ and $\mathbf{S}_y = [e_{y,1}, \ldots, e_{y,i}, \ldots, e_{y,J}]$ respectively, where $e_{x,i}, e_{y,i} \in \mathbb{R}^d$ are the embeddings of the i-th word of S_x and S_y respectively. $\mathbf{S}_x$ and $\mathbf{S}_y$ are used to create three similarity matrices, each of which is regarded as an input channel of a CNN. CNN extracts high level features from the similarity matrices. All features are finally concatenated and synthesized by a multilayer perceptron (MLP) to form a matching score.

Specifically, in channel one, $\forall i, j$, element $e_{1,i,j}$ in similarity matrix $\mathbf{M}_1$ is calculated by

$$e_{1,i,j} = h(e_{x,i}^{\intercal} \cdot e_{y,j}), \tag{3}$$

where $h(\cdot)$ could be ReLU or tanh. $\mathbf{M}_1$ matches S_x and S_y on words.

In channel two, we employ bidirectional gated recurrent units (BiGRU) (Chung et al. 2014) to encode S_x and S_y into hidden vectors. A BiGRU consists of a forward RNN and a backward RNN. The forward RNN processes S_x as it is ordered (i.e., from $e_{x,1}$ to $e_{x,I}$), and generates a sequence of hidden states $(\overrightarrow{h}_1, \ldots, \overrightarrow{h}_I)$. The backward RNN reads S_x in its reverse order (i.e., from $e_{x,I}$ to $e_{x,1}$) and generates a sequence of backward hidden states $(\overleftarrow{h}_1, \ldots, \overleftarrow{h}_I)$. BiGRU then forms the hidden vectors of S_x as $\{h_{x,i} = [\overrightarrow{h}_i, \overleftarrow{h}_i]\}_{i=1}^{I}$ by concatenating the forward and the backward hidden states. More specifically, $\forall i$, $\overrightarrow{h}_i \in \mathbb{R}^m$ is calculated by

$$\begin{aligned}
z_i &= \sigma(W_z e_{x,i} + U_z \overrightarrow{h}_{i-1}) \\
r_i &= \sigma(W_r e_{x,i} + U_r \overrightarrow{h}_{i-1}) \\
\widetilde{h}_i &= tanh(W_h e_{x,i} + U_h (r_i \odot \overrightarrow{h}_{i-1})) \\
\overrightarrow{h}_i &= z_i \odot \widetilde{h}_i + (1 - z_i) \odot \overrightarrow{h}_{i-1},
\end{aligned} \tag{4}$$

where $\overrightarrow{h}_0 = 0$, z_i and r_i are an update gate and a reset gate respectively, and W_z, W_h, W_r, U_z, U_r,U_h are parameters. The backward hidden state $\overleftarrow{h}_i \in \mathbb{R}^m$ is obtained in a similar way. Following the same procedure, we get $\{h_{y,i}\}_{i=1}^{J}$ as the hidden vectors of S_y. $\forall i, j$, we calculate element $e_{2,i,j}$ in similarity matrix $\mathbf{M}_2$ by

$$e_{2,i,j} = h(h_{x,i}^{\intercal} W_2 h_{y,j} + b_2), \tag{5}$$

where $W_2 \in \mathbb{R}^{2m\times 2m}$ and $b_2 \in \mathbb{R}$ are parameters. BiGRU models dependencies among words and encodes sequential information of text into hidden vectors, therefore, $\mathbf{M}_2$ matches S_x and S_y on local structures (i.e., sequential structures such as phrases and clauses) of text.

In the last channel, we employ another BiGRU to process the sequences of S_x and S_y which consists of the knowledge enhanced representations in Equation (2), and obtain the knowledge enhanced hidden states $\mathbf{kh}_x = (kh_{x,1}, \ldots, kh_{x,I})$ and $\mathbf{kh}_y = (kh_{y,1}, \ldots, kh_{y,J})$ for S_x and S_y respectively. Similar to channel two, $\forall i, j$, element $e_{3,i,j}$ in similarity matrix $\mathbf{M}_3$ is given by

$$e_{3,i,j} = h(kh_{x,i}^{\intercal} \cdot W_3 \cdot kh_{y,j} + b_3), \tag{6}$$

where $W_3 \in \mathbb{R}^{2m\times 2m}$ and $b_3 \in \mathbb{R}$ are parameters. Prior knowledge represents a kind of global semantics of S_x and S_y, and therefore $\mathbf{M_3}$ matches S_x and S_y on global context of text. Note that channel three also models the sequential structures and the dependencies in text. The difference is that the structures and the dependencies are modeled under the supervision of the prior knowledge.

The similarity matrices are then processed by a CNN to abstract high level features. $\forall i = 1, 2, 3$, CNN regards a similarity matrix as an input channel, and alternates convolution and max-pooling operations. Suppose that $z^{(l,f)} = \left[z_{i,j}^{(l,f)}\right]_{I^{(l,f)}\times J^{(l,f)}}$ denotes the output of feature maps of type-f on layer-l, where $z^{(0,f)} = \mathbf{M}_f, \forall f = 1, 2, 3$. On convolution layers (i.e. $\forall l = 1, 3, 5, \ldots,$), we employ a 2D convolution operation with a window size $r_w^{(l,f)} \times r_h^{(l,f)}$, and define $z_{i,j}^{(l,f)}$ as

$$z_{i,j}^{(l,f)} = \sigma(\sum_{f'=0}^{F_{l-1}} \sum_{s=0}^{r_w^{(l,f)}} \sum_{t=0}^{r_h^{(l,f)}} \mathbf{w}_{s,t}^{(l,f)} \cdot z_{i+s,j+t}^{(l-1,f')} + b^{l,k}), \tag{7}$$

where $\sigma(\cdot)$ is a ReLU, and $\mathbf{w}^{(l,f)} \in \mathbb{R}^{r_w^{(l,f)}\times r_h^{(l,f)}}$ and $b^{l,k}$ are parameters of the f-th feature map on the l-th layer, and F_{l-1} is the number of feature maps on the $(l-1)$-th layer. We employ a max pooling operation after a convolution operation and can be formulated as

$$z_{i,j}^{(l,f)} = \max_{p_w^{(l,f)} > s \geq 0} \max_{p_h^{(l,f)} > t \geq 0} z_{i+s,j+t}, \forall l = 2, 4, 6, \ldots, \tag{8}$$

where $p_w^{(l,f)}$ and $p_h^{(l,f)}$ are the width and the height of the 2D pooling respectively.

The output of the final feature maps are concatenated as a vector v and fed to a two-layer feed-forward neural network (i.e., MLP) to calculate a matching score $g(S_x, S_y)$:

$$g(S_x, S_y) = \sigma_1\left(\mathbf{w}_2^{\intercal} \cdot \sigma_2\left(\mathbf{w}_1^{\intercal} v + b_4\right) + b_5\right), \tag{9}$$

where $\mathbf{w}_1$, $\mathbf{w}_2$, b_4, and b_5 are parameters. $\sigma_1(\cdot)$ is softmax and $\sigma_2(\cdot)$ is tanh.

KEHNN matches two objects by letting them meet at the beginning, which is the advantage of 2D CNN (Hu et al. 2014) over other methods. Moreover, it constructs interaction matrices by considering multiple levels of semantics (words, local structures, and global context). Therefore semantic relations between the two objects can be sufficiently modeled and leveraged in building the matching function. Our model extends the existing models (Hu et al. 2014; Wan et al. 2015) by fusing extra knowledge into matching and conducting matching with multiple channels.

We learn $g(\cdot,\cdot)$ by minimizing cross entropy (Levin and Fleisher 1988) with $\mathcal{D}$ and $\left\{\cup_{i=1}^{N}\mathbf{k}_{x,i}, \cup_{i=1}^{N}\mathbf{k}_{y,i}\right\}$. Let Θ denote the parameters of our model. Then the objective function of learning can be formulated as

$$\mathcal{L}(\mathcal{D};\Theta) = -\sum_{i=1}^{N}\sum_{c=1}^{C} P_c^g(l_i) \cdot log(P_c(g(S_{x,i}, S_{y,i})), \tag{10}$$

where N in the number of instances in $\mathcal{D}$, and C is the number of values of labels in $\mathcal{D}$. $P_c(g(S_{x,i}, S_{y,i}))$ returns the c-th element from the C-dimensional vector $g(S_{x,i}, S_{y,i})$, and $P_c^g(l_i)$ is 1 or 0, indicating whether l_i equals to c or not. We optimize the objective function using back-propagation and the parameters are updated by stochastic gradient descent with Adam algorithm (Kingma and Ba 2014). As regularization, we employ early-stopping (Lawrence and Giles 2000) and dropout (Srivastava et al. 2014) with rate of 0.5. We set the initial learning rate and the batch size as 0.01 and 50 respectively.

Prior Knowledge Acquisition

Prior knowledge plays a key role to the success of our model. As described above, we expect prior knowledge to represent global context of input. In practice, we can use tags, keywords, topics, or entities (with meta information from a knowledge base) that are related to the input as instantiation of the prior knowledge. Such prior knowledge could be obtained either from the metadata of the input, or from extra algorithms. Algorithms include tag recommendation (Wu et al. 2016), keyword extraction (Wu et al. 2015), topic modeling (Blei, Ng, and Jordan 2003) and entity linking (Han, Sun, and Zhao 2011) can be utilized to extract the prior knowledge from multiple resources like web documents, social media and knowledge base, and embedding techniques (Pennington, Socher, and Manning 2014; Wang et al. 2014) can be used to transform the raw prior knowledge to vectors as $\mathbf{k}_x$ and $\mathbf{k}_y$.

In our experiments, we used question tags which represent a kind of metadata in CQA as the prior knowledge in the QA task. For the conversation task, we pre-trained a Twitter LDA model (Zhao et al. 2011) with external large scale social media data and took the topics predicted by LDA as the prior knowledge, as the topics could help us group text with similar meaning. Both the tags and the topics are transformed as vectors by the word2vec algorithm (Pennington, Socher, and Manning 2014) (see our experiments) and used in matching through the knowledge gates. The tags and the topics represent a high level abstraction from human or an automatic algorithm to the QA pairs or the message-response pairs, and therefore, they can reflect the global semantics of the input of the two tasks. We leave the discussion of other types of prior knowledge (e.g., entities) as future work.

Experiments

We tested our model on two matching tasks: answer selection and response selection.

Baseline

We consider the following models as baselines:

Basic models: each piece of text is first embedded to a vector space by neural networks, and then their matching degree is computed by the two vectors. The embedding methods include multi-layer perceptron (MLP), LSTM (Lowe et al. 2015), and CNN (Arc1) (Hu et al. 2014). Besides these, we also implemented DeepMatch$_{topic}$ proposed in (Lu and Li 2013), and the convolution neural tensor network (CNTN) (Qiu and Huang 2015) proposed for community question answering. We also use cosine score with one hot representations (tf·idf) as inputs to compute the matching degree.

Attentive matching models: we selected Attentive-LSTM proposed by Tan et al. (2015) and Match-LSTM proposed by Wang et al. (2015) as baselines which represent the state-of-the-art matching approaches with attention mechanism.

2D matching models: we implemented Arc2 (Hu et al. 2014), MatchPyramid (Pang et al. 2016), and MV-LSTM (Wan et al. 2015). In addition to these models, we also compared our model with MultiGranCNN (Yin and Schütze 2015) which also ensures multigranular comparability but does not consider prior knowledge.

Knowledge augmented models: we implemented the model proposed by (Severyn and Moschitti 2015) where prior knowledge is the same as our model but used as additional features and denote it as AddFeature.

We implemented all baselines and KEHNN by Theano (Theano Development Team 2016). For all models, we set the dimensionality of word embedding (i.e., d) as 100 and the maximum text length (i.e., I and J) as 200. In LSTM based models and BiGRU in our model, we set the dimensionality of hidden states as 100 (i.e., m). We only used one convolution layer and one max-pooling layer in all CNN based models, because we found that the performance of the models did not get better with the number of layers increased. For Arc2, MatchPyramid, MV-LSTM, and KEHNN, we tuned the window size in convolution and pooling in $\{(2,2),(3,3)(4,4)\}$ and chose $(3,3)$ finally. The number of feature maps is 8. For Arc1 and CNTN, we selected the window size from $\{2,3,4\}$ and set it as 3 finally. The number of feature maps is 200. In MLP, we tuned the dimensionality of the hidden layer in $\{50,200,400,800\}$ and set it as 50 finally. We implemented MultiGranCNN and AddFeature following the settings in the existing literatures. S_x and S_y in KEHNN shared word embeddings, knowledge embeddings, parameters of BiGRUs, and parameters of the knowledge gates. All tuning was conducted on validation sets. The activation functions in baselines are the same as those in our model. We shared the code of our model at *an anonymous url for blind review*.

Data	#question	#answer	#answers per question
Training	2600	16541	6.36
Dev	300	1645	5.48
Test	329	1976	6.00

Table 2: Statistics of the QA data set

Answer Selection

The goal of the task is to select a good answer from answer candidates for a question. We used a public data set in SemEval 2015 (AlessandroMoschitti, Glass, and Randeree 2015), which collects question-answer pairs from Qatar Living Forum[2] and requires to classify the answers into 3 categories (i.e. $C = 3$ in our model) including good, potential and bad. The ratio of the three categories is 51:10:39. The statistics of the data is summarized in Table 2. We used classification accuracy as an evaluation metric.

Specific Setting In this task, we regarded question tags assigned by askers as prior knowledge (both $\mathbf{k}_x$ and $\mathbf{k}_y$). There are 27 tags in the Qatar Living data. Knowledge vector $\mathbf{k}$ was initialized by averaging the embeddings of words in the tag. For all baselines and our model, the word embedding, the idf table and the topic model (in DeepMatch$_{topic}$) were trained on a Qatar living raw text provided by SemEval 2015 [3]. We fixed the word embedding during the training process as the training data is small, and set h in Equation (3), (5), (6) as ReLU. The reason of using ReLU here is that ReLU is able to prevent overfitting when the question-answer pairs are not enough.

Results JAIST, the champion of the task in SemEval 2015, used 12 features and an SVM classifier and achieved an accuracy of 0.725. TFIDF performs very bad on this task, as it only matches QA pairs by word overlap. From Table 3, we can see that advanced neural networks, such as CNTN, MV-LSTM, Attentive-LSTM, Match-LSTM, and KEHNN, outperform JAIST's model, indicating that hand-crafted features are less powerful than deep learning methods. Models that match text pairs by interaction representations like Arc2 and MatchPyramid are not better than models that perform matching with sentence embeddings like Arc1. This is because the training data is small and we fixed the word embedding in learning. LSTM based models in general performs better than CNN based models, because they can capture sequential information and dependencies in text. KEHNN outperforms all baseline methods, and the improvement is statistically significant (t-test with p-value ≤ 0.01).

Response Selection

The goal of response selection is to select a proper response for a message from a candidate pool in a conversational environment. We used a public English conversation data set,

[2] http://www.qatarliving.com/forum

[3] http://alt.qcri.org/semeval2015/task3/index.php?id=data-and-tools

	ACC
TFIDF	0.456
MLP	0.713
DeepMatch$_{topic}$ (Lu and Li 2013)	0.682
Arc1 (Hu et al. 2014)	0.715
CNTN (Huang et al. 2013)	0.735
LSTM (Lowe et al. 2015)	0.725
Attentive-LSTM (Tan, Xiang, and Zhou 2015)	0.736
Match-LSTM (Wang and Jiang 2015)	0.743
Arc2 (Hu et al. 2014)	0.715
AddFeature (Severyn and Moschitti 2015)	0.743
MatchPyramid (Pang et al. 2016)	0.717
MultiGranCNN (Yin and Schütze 2015)	0.743
MV-LSTM (Wan et al. 2015)	0.735
JAIST (Tran et al. 2015)	0.725
KEHNN	**0.753**

Table 3: Results on answer selection

the Ubuntu Corpus (Lowe et al. 2015), to conduct the experiment. The corpus consists of a large number of human-human dialogues about Ubuntu technique. Each dialogue contains at least 3 turns, and we only kept the last two utterances as we study text pair matching and ignore context information. We used the data pre-processed by Xu et al. (Xu et al. 2016)[4], in which all urls and numbers were replaced by special placeholders. The training set contains 1 million message-response pairs with a ratio 1:1 between positive and negative responses, and both the validation set and the test set have 0.5 million message-response pairs with a ratio 1:9 between positive and negative responses. We followed Lowe et al. (Lowe et al. 2015) and employed recall at position k in n candidates as evaluation metrics and denoted the metrics as $R_n@k$.

Specific Setting In this task, we trained a topic model to generate topics for both messages and responses as prior knowledge. We crawled 8 million questions (question and description) from the "Computers & Internet" category in Yahoo! Answers, and utilized these data to train a Twitter LDA model (Zhao et al. 2011) with 100 topics. In order to construct $\mathbf{k}_{x,i}$ and $\mathbf{k}_{y,i}$, we separately assigned a topic to a message and a response by the inference algorithm of Twitter LDA. Then we transformed the topic to a vector by averaging the embeddings of top 20 words under the topic. Word embedding tables were initialized using the public word vectors available at `http://nlp.stanford.edu/projects/glove` (trained on Twitter) and updated in learning. Tanh is used as h in Equation (3), (5), (6). The idf table is calculated using the training data.

Results Table 4 reports the evaluation results on response selection. Our method outperforms baseline models on all metrics, and the improvement is statistically significant (t-test with p-value ≤ 0.01). In the data set, as the training data becomes large and we updated word embedding in learning, Arc2 and MatchPyraimd are much better than Arc1. LSTM based models perform better than CNN based models, which

[4]`https://www.dropbox.com/s/2fdn26rj6h9bpvl/ubuntudata.zip?dl=0`

	R_2@1	R_{10}@1	R_{10}@2	R_{10}@5
TFIDF	0.681	0.383	0.482	0.686
MLP	0.651	0.256	0.380	0.703
DeepMatch$_{topic}$	0.593	0.345	0.376	0.693
Arc1	0.665	0.221	0.360	0.684
CNTN	0.743	0.349	0.512	0.797
LSTM	0.725	0.361	0.494	0.801
Attentive-LSTM	0.758	0.381	0.545	0.801
Match-LSTM	0.685	0.289	0.430	0.701
Arc2	0.736	0.380	0.534	0.777
AddFeature	0.763	0.425	0.570	0.804
MatchPyramid	0.743	0.420	0.554	0.786
MultiGranCNN	0.762	0.436	0.571	0.792
MV-LSTM	0.767	0.410	0.565	0.800
KEHNN	**0.786**	**0.460**	**0.591**	**0.819**

Table 4: Evaluation results on response selection

	Conversation				QA
	R_2@1	R_{10}@1	R_{10}@2	R_{10}@5	ACC
$\mathbf{M}_1$	0.743	0.420	0.554	0.786	0.717
$\mathbf{M}_2$	0.779	0.425	0.565	0.800	0.734
$\mathbf{M}_3$	0.750	0.360	0.531	0.791	0.738
$\mathbf{M}_1$+$\mathbf{M}_2$	0.781	0.435	0.578	0.810	0.745
Remove$_g$	0.778	0.433	0.580	0.810	0.746
Concat	0.784	0.445	0.583	0.817	0.747
KEHNN	0.786	0.460	0.591	0.819	0.753

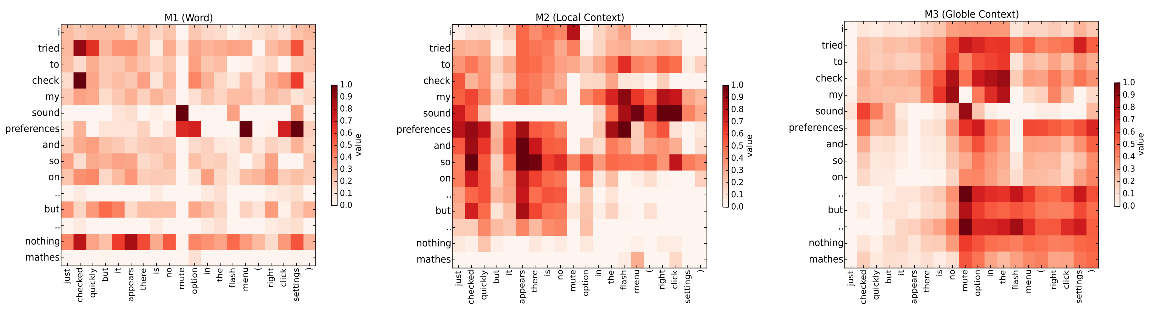
Table 5: Model Ablation. Concat denotes $\mathbf{M}_1$ and $\mathbf{M}_2$ with prior knowledge as an additional feature

is consistent with the results in the QA task.

Discussions

Model ablation: we first analyze the effect of different parts of KEHNN. Table 5 gives the results. First, $\mathbf{M}_3$ is the best single channel on the QA data, but it is worse than $\mathbf{M}_2$ on the conversation data. This is because there is noise in the automatically generated topics as the prior knowledge. In spite of this, when erasing $\mathbf{M}_3$ from KEHNN (denoted as $\mathbf{M}_1$+$\mathbf{M}_2$) or removing the knowledge gates (denoted as Remove$_g$) from $\mathbf{M}_3$ but the keeping three channels, the performance of the model dropped. The results demonstrate the importance of the prior knowledge to matching: it can improve matching accuracy and cannot be simply replaced by more parameters. Moreover, we also compared KEHNN with Concat which keeps $\mathbf{M}_1$ and $\mathbf{M}_2$ but uses the same prior knowledge as additional features like AddFeature. Although Concat further gets closer to KEHNN, there are still clear gaps on R_{10}@1, R_{10}@2, and ACC. The results verified the advantage of the knowledge gate over the heuristic method in terms of leveraging prior knowledge for matching. Different channels provide different information, and the information is complementary to each other. This explains why $\mathbf{M}_1$+$\mathbf{M}_2$ is better than $\mathbf{M}_1$, and KEHNN is further better than $\mathbf{M}_1$+$\mathbf{M}_2$.

Performance across length : we then examine if KEHNN can improve matching accuracy on long text by binning text pairs in the two data sets into 4 buckets according to the length of the concatenation of the text. Table 6 compares KEHNN with LSTM, Attentive-LSTM, MV-



(a) $\mathbf{M}_1$ (Word) (b) $\mathbf{M}_2$ (Local Context) (c) $\mathbf{M}_3$ (Global Context)

Figure 2: Model visualization. The darker areas mean larger values.

(a) Accuracy on QA dataset

Length	[0, 30)	[30, 60)	[60, 90)	[90, ∞)
LSTM	0.768	0.705	0.728	0.726
Attentive-LSTM	0.792	0.719	0.746	0.732
MV-LSTM	0.788	0.708	0.746	0.739
MultiGranCNN	0.787	0.694	0.712	0.712
$\mathbf{M}_1$+$\mathbf{M}_2$	0.787	0.719	0.760	0.731
KEHNN	0.792	0.721	0.765	0.746

(b) R_{10}@1 on Ubuntu dataset

Length	[0, 10)	[10, 20)	[20, 30)	[30, ∞)
LSTM	0.367	0.361	0.345	0.316
Attentive-LSTM	0.383	0.381	0.341	0.318
MV-LSTM	0.416	0.410	0.390	0.369
MultiGranCNN	0.441	0.433	0.431	0.399
$\mathbf{M}_1$+$\mathbf{M}_2$	0.443	0.428	0.415	0.410
KEHNN	0.464	0.445	0.452	0.437

Table 6: Performance on different length of text

LSTM, MultiGranCNN and $\mathbf{M}_1$+$\mathbf{M}_2$. We can see that on both data sets, the performance becomes worse as the text becomes longer, and KEHNN can slow the performance drop. From the comparison with $\mathbf{M}_1$+$\mathbf{M}_2$, we can see that the prior knowledge contributes a lot on text longer than 90 words on the QA data, and is useful from short text to long text in the conversation data.

Visualization: we finally visualize $\mathbf{M}_1$, $\mathbf{M}_2$, and $\mathbf{M}_3$ in Figure 2 with an example from the Ubuntu Corpus . The example is *"message: i tried to check my sound preferences and so on .. but .. nothing matches"* and *"response: just checked quickly but it appears there is no mute option in the flash menu (right click settings)."*. Top words under the topic (assigned by LDA) of this examples includes *sound, audio ...* . Similar word pairs like (sound,mute) and segment pairs like "my sound preference" and "flash menu (right click setting)" have large values in $\mathbf{M}_1$ and $\mathbf{M}_2$ respectively. A major difference between $\mathbf{M}_2$ and $\mathbf{M}_3$ is that text after "mute" in the response is more important in $\mathbf{M}_3$. This is because the prior knowledge (topics) is about "audio" and highlighted the text segment in matching as it is more related to the topics.

Conclusion

This paper proposes KEHNN which can leverage prior knowledge in semantic matching. Experiment results show that the model can significantly outperform state-of-the-art matching models on two data sets, and especially improve matching accuracy on long text. We are going to study how to use different prior knowledge in future.

Acknowledgment

We appreciate valuable comments provided by anonymous reviewers and our discussions with Zhao Yan. Yu Wu is supported by Microsoft Fellowship Scholarship and AdeptMind Scholarship. This work was supported by the National Natural Science Foundation of China (Grand Nos. 61672081, U1636211, 61370126), Beijing Advanced Innovation Center for Imaging Technology (No.BAICIT-2016001), National High Technology Research and Development Program of China (No.2015AA016004), and the Fund of the State Key Laboratory of Software Development Environment (No.SKLSDE-2015ZX-16).

References

AlessandroMoschitti, P. L. W.; Glass, J.; and Randeree, B. 2015. Semeval-2015 task 3: Answer selection in community question answering. *SemEval-2015* 269.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *the Journal of machine Learning research* 3:993–1022.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Dolan, B.; Quirk, C.; and Brockett, C. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *ACL*, 350.

Ghosh, S.; Vinyals, O.; Strope, B.; Roy, S.; Dean, T.; and Heck, L. 2016. Contextual lstm (clstm) models for large scale nlp tasks. *arXiv preprint arXiv:1602.06291*.

Han, X.; Sun, L.; and Zhao, J. 2011. Collective entity linking in web text: a graph-based method. In *SIGIR*, 765–774. ACM.

He, H., and Lin, J. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of NAACL-HLT*, 937–948.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hu, B.; Lu, Z.; Li, H.; and Chen, Q. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*, 2042–2050.

Huang, P.-S.; He, X.; Gao, J.; Deng, L.; Acero, A.; and Heck, L. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, 2333–2338. ACM.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lawrence, S., and Giles, C. L. 2000. Overfitting and neural networks: conjugate gradient and backpropagation. In *Neural Networks, 2000. IJCNN 200*, volume 1, 114–119. IEEE.

Levin, E., and Fleisher, M. 1988. Accelerated learning in layered neural networks. *Complex systems* 2:625–640.

Liu, P.; Qiu, X.; Chen, J.; and Huang, X. 2016. Deep fusion lstms for text semantic matching. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Liu, P.; Qiu, X.; and Huang, X. 2016. Modelling interaction of sentence pair with coupled-lstms. *arXiv preprint arXiv:1605.05573*.

Lowe, R.; Pow, N.; Serban, I.; and Pineau, J. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Lu, Z., and Li, H. 2013. A deep architecture for matching short texts. In *NIPS*, 1367–1375.

Pang, L.; Lan, Y.; Guo, J.; Xu, J.; Wan, S.; and Cheng, X. 2016. Text matching as image recognition.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, 1532–43.

Qiu, X., and Huang, X. 2015. Convolutional neural tensor network architecture for community-based question answering. In *IJCAI*, 1305–1311.

Rocktäschel, T.; Grefenstette, E.; Hermann, K. M.; Kočiskỳ, T.; and Blunsom, P. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Severyn, A., and Moschitti, A. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*, 373–382. ACM.

Shen, Y.; He, X.; Gao, J.; Deng, L.; and Mesnil, G. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *CIKM*, 101–110. ACM.

Socher, R.; Huang, E. H.; Pennin, J.; Manning, C. D.; and Ng, A. Y. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, 801–809.

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*, 926–934.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Tan, M.; Xiang, B.; and Zhou, B. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688.

Tran, Q. H.; Tran, V.; Vu, T.; Nguyen, M.; and Bao Pham, S. 2015. Jaist: Combining multiple features for answer selection in community question answering. In *(SemEval 2015)*, 215–219. Denver, Colorado: Association for Computational Linguistics.

Voorhees, E. M., et al. 1999. The trec-8 question answering track report. In *Trec*, volume 99, 77–82.

Wan, S.; Lan, Y.; Guo, J.; Xu, J.; Pang, L.; and Cheng, X. 2015. A deep architecture for semantic matching with multiple positional sentence representations. *arXiv preprint arXiv:1511.08277*.

Wan, S.; Lan, Y.; Xu, J.; Guo, J.; Pang, L.; and Cheng, X. 2016. Match-srnn: Modeling the recursive matching structure with spatial rnn. *arXiv preprint arXiv:1604.04378*.

Wang, S., and Jiang, J. 2015. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*.

Wang, H.; Lu, Z.; Li, H.; and Chen, E. 2013. A dataset for research on short-text conversations. In *EMNLP*, 935–945.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 1112–1119. Citeseer.

Wang, B.; Liu, K.; and Zhao, J. 2016. Inner attention based recurrent neural networks for answer selection. In *ACL*.

Wu, Y.; Wu, W.; Li, Z.; and Zhou, M. 2015. Mining query subtopics from questions in community question answering. In *AAAI*, 339–345.

Wu, Y.; Wu, W.; Li, Z.; and Zhou, M. 2016. Improving recommendation of tail tags for questions in community question answering. In *AAAI*.

Xu, Z.; Liu, B.; Wang, B.; Sun, C.; and Wang, X. 2016. Incorporating loose-structured knowledge into lstm with recall gate for conversation modeling. *arXiv preprint arXiv:1605.05110*.

Yin, W., and Schütze, H. 2015. Multigrancnn: An architecture for general matching of text chunks on multiple levels of granularity. In *ACL*, 63–73.

Yin, W.; Schütze, H.; Xiang, B.; and Zhou, B. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Zhao, W. X.; Jiang, J.; Weng, J.; He, J.; Lim, E.-P.; Yan, H.; and Li, X. 2011. Comparing twitter and traditional media using topic models. In *Advances in Information Retrieval*. Springer. 338–349.