The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Improved Text Matching by Enhancing Mutual Information

Yang Liu,[1,2] **Wenge Rong,**[1,2] **Zhang Xiong**[1,2]
[1]State Key Laboratory of Software Development Environment, Beihang University, China
[2]School of Computer Science and Engineering, Beihang University, China
{liuyang1123, w.rong, xiongz}@buaa.edu.cn

Abstract

Text matching is a core issue for question answering (QA), information retrieval (IR) and many other fields. We propose to reformulate the original text, i.e., generating a new text that is semantically equivalent to original text, to improve text matching degree. Intuitively, the generated text improves mutual information between two text sequences. We employ the generative adversarial network as the reformulation model where there is a discriminator to guide the text generating process. In this work, we focus on matching question and answers. The task is to rank answers based on QA matching degree. We first reformulate the original question without changing the asker's intent, then compute a relevance score for each answer. To evaluate the method, we collected questions and answers from Zhihu. In addition, we also conduct substantial experiments on public data such as SemEval and WikiQA to compare our method with existing methods. Experimental results demonstrate that after adding the reformulated question, the ranking performance across different matching models can be improved consistently, indicating that the reformulated question has enhanced mutual information and effectively bridged the semantic gap between QA.

1 Introduction

Text matching is of great importance for many applications. The matching methods vary from traditional words matching to modern semantic matching . Various deep matching models (Severyn and Moschitti 2015; Tymoshenko and Moschitti 2015; Bogdanova et al. 2017; Yang et al. 2016) have been proposed in recent years. Those models can be divided into representation-focused methods and interaction-focused methods. The representation-focused methods (Hu et al. 2014; Jansen, Surdeanu, and Clark 2014; Tan et al. 2016) aim to learn a good latent representation for a text sequence and then conduct matching between the two representations. The interaction-focused (Bogdanova et al. 2017; Wang and Jiang 2017) methods could account for lexical interactions while learning text representations, a typical instance is the attention mechanism (Bahdanau, Cho, and Bengio 2015). In the view of information theory, those matching models can be regarded as a kind of information channel between question answering (QA). For QA that has close relationship, they share more mutual information. A better matching model means a better information channel, thus uncovering more mutual information between QA. Most prior work (Tan et al. 2016; Wang and Jiang 2017) tries to build a better matching model to enhance mutual information. We, instead, enhance mutual information by generating extra text, i.e., reformulating the original text in another expression.

To be specific, we focus on matching QA and adopt the generative adversarial network (GAN) (Goodfellow et al. 2014) framework to rewrite questions in an unsupervised way. In the GAN framework, a generator consisting of recurrent neural networks (RNN) is used to generate new question. Meanwhile, a discriminator is trained to force the new question to be semantically close to original question in embedding space. The rewriting process is similar to monolingual machine translation where the rewriting question is translated from original question. A main problem for the generator is the non-differentiability of discrete text. Since each word of the generated question is derived from the $argmax$ (non-differentiable) operation on the probability distribution over vocabulary, the predicted word is non-differentiable w.r.t. generator parameters. Thus, the generator parameters cannot be updated in back propagation. To address this issue, we adopt the policy gradients (Williams 1992; Sutton et al. 1999), which is usually used in teaching machine playing games. We use policy gradients from the discriminator to guide the updation of generator parameters so that the generator can generate more plausible samples. Experiments are conducted on Zhihu, WikiQA and SemEval datasets and the results demonstrate that the rewriting module improves ranking performance consistently.

2 Related Work

Deep Matching Models. In recent years, neural networks have shown great superiority in learning sequences' semantic representation and have achieved a great success in a variety of NLP tasks as well as computer vision and speech recognition tasks. In the QA matching domain, (Wang and Nyberg 2015) explores a stacked bidirectional LSTM (Hochreiter and Schmidhuber 1997) to capture context information for the current representation of words. In parallel with recurrent neural networks, convolutional neural networks (CNNs) are also used to learn representations of sequence (Yu et al. 2014). Many studies have shown that CNNs are significantly faster than RNNs due to the parallel execution architecture of GPU. To utilize the merits of both RNNs and CNNs, (Tan et al. 2016) investigates various combination of CNNs and LSTM architectures. (Yin et al. 2016) further introduces the attention mechanism (Bahdanau, Cho, and Bengio 2015) into CNNs to learn to emphasis on different words in sequence. (Wang and Jiang 2017) proposes to perform word-level matching and aggregate the word-level similarities using various distance functions for classification.

GAN. GAN (Goodfellow et al. 2014) is a popular generative model that has been successfully applied in various computer vision tasks and it can generate high-quality plausible pictures. Some work (Reed et al. 2016; Zhang et al. 2016) has used GAN to generate picture from text, while using GAN to generate text is less explored due to the non-differentiability of discrete words. Recently, (Yu et al. 2017) has adopted GAN to generate poems and music, which sounds intersting. In each time step, they apply Monte Carlo search to restrain the next generating action.

3 Proposed Approach

3.1 Question Rewriting

We propose to rewrite question via adversarial training. In particular, a generator G is responsible for generating text and tries to confuse a discriminator D which is trained to distinguish whether the input is real or synthetic. In adversarial training, the discriminator and generator compete with each other, the discriminator D force the generator G to produce question that is indistinguishable to the real question in embedding space. D accomplishes that by propagating policy gradients back to G.

Preliminary GAN was first introduced by (Goodfellow et al. 2014) to generate plausible images. GAN consists of a generator $G\left(x;\theta_g\right)$ and a discriminator $D\left(y;\theta_d\right)$. The goal is to train an unsupervised deep generative model that can generate samples y(the distribution denoted as P_{θ_g}) that are as real as the true samples x(the distribution denoted as P_{data}). After many training iterations, G and D reach an equilibrium state where G can successfully (by generating plausible samples) fool D and D can't distinguish whether a sample is real or fake. The training process is a minimax game with the value function $V\left(D,G\right)$:

$$\min_G \max_D V\left(D,G\right) = \mathbb{E}_{x\sim P_{data}}\left[\log D\left(x\right)\right] + \mathbb{E}_{y\sim P_{\theta_g}}\left[\log(1-D\left(y\right))\right] \tag{1}$$

where $y=G\left(x;\theta_g\right)$ is the generating samples of G conditioned on x. In Eq. 1, we want to classify x and y as positive (real) and negative (fake) respectively. For image generating task, y is always generated from a random noise variable $z\in\mathbb{R}^d$. In this work, y is generated from x and the generator employs the encoder-decoder (Sutskever, Vinyals, and Le 2014) architecture, G first encodes x and uses it to predict each word of the rewriting question. The training objective for G is to maximize the probability of D making a mistake while D tries its best to correctly classify the inputs. More specifically, G wants to fool D so G wants D to classify generated samples y to be real, i.e., $D(y)$ approaches to 1(real). On the other side, D tries its best to correctly classify samples, i.e., $D(y)$ approaches to 0(fake) and $D(x)$ approaches to 1. According to above rules, the parameters θ_g and θ_d are updated by two optimizers as following:

$$\theta_g \leftarrow \theta_g - \alpha\nabla_{\theta_g}\mathbb{E}_{y\sim P_{\theta_g}}\log(1-D(y)) \tag{2}$$

$$\begin{aligned}\theta_d \leftarrow \theta_d &+ \alpha\nabla_{\theta_d}\mathbb{E}_{x\sim P_{data}}\log D(x)\\ &+ \alpha\nabla_{\theta_d}\mathbb{E}_{y\sim P_{\theta_g}}\log(1-D(y))\end{aligned} \tag{3}$$

where α is the learning rate. It is worth noting that θ_g can also be updated by $\theta_g \leftarrow \theta_g + \alpha\nabla_{\theta_g}\mathbb{E}_{y\sim P_{\theta_g}}\log D(y)$. For image generation, y is a picture and y is differentiable w.r.t. θ_g. For text generation, however, y consists of discrete words and is no more differentiable w.r.t. θ_g since y is the outcome of $argmax$ function, which is non-differentiable.

Generating Model The question generating architecture is shown in Fig. 1. In the forward pass, G first encodes the question into m hidden states where m is the length of question. When predicting a word, the system learns a distinct representation of input question using attention mechanism (Bahdanau, Cho, and Bengio 2015).

The input question is encoded using bidirectional RNNs (Schuster and Paliwal 1997) that compose of a forward RNN and a backward RNN. The forward RNN reads question in normal order $(x_1, x_2, ...x_m)$ and outputs hidden states $\left(\overrightarrow{h_1}, \overrightarrow{h_2}, ...\overrightarrow{h_m}\right)$. The backward RNN reads question in reverse order $(x_m, ...x_2, x_1)$ and outputs hidden states $\left(\overleftarrow{h_m}, \overleftarrow{h_{m-1}}...\overleftarrow{h_1}\right)$. The final outputs are concatenations of forward and backward states, i.e., $h_i = \left[\overrightarrow{h_i}, \overleftarrow{h_i}\right], 1\le i\le m$. Thus, each hidden state h_i contains context information of both preceding words and succeeding words information with a focus on the i-th word.

In generating phase, a decoder RNN_{dec} is responsible to estimate the joint probability over predicted words:

$$G(y|x) = \prod_{t=1}^{m'} p(y_t|y_{<t}, x) \tag{4}$$

For each predicted word y_t, RNN_{dec} outputs a probability distribution $p_t\in\mathbb{R}^V$ (V is the vocabulary size) over the vocabulary:

$$s_t = RNN_{dec}\left(s_{t-1}, y_{t-1}, c_t\right) \tag{5}$$

$$p\left(y_t|y_{<t}, x\right) = softmax\left(Us_t + b\right) \tag{6}$$

$$y_t = \arg\max\left(p\left(y_t|y_{<t}, x\right)\right) \tag{7}$$

where s_t is hidden state of RNN_{dec} at time step t. c_t is a distinct representation of source input computed specifically for y_t:

$$c_t = \sum\nolimits_{i=1}^{m}\alpha_{t,i}h_i$$

$$\alpha_{t,i} = \exp\left(h_i^TWs_{t-1}\right)/\sum\nolimits_{j=1}^{m}\exp\left(h_j^TWs_{t-1}\right)$$

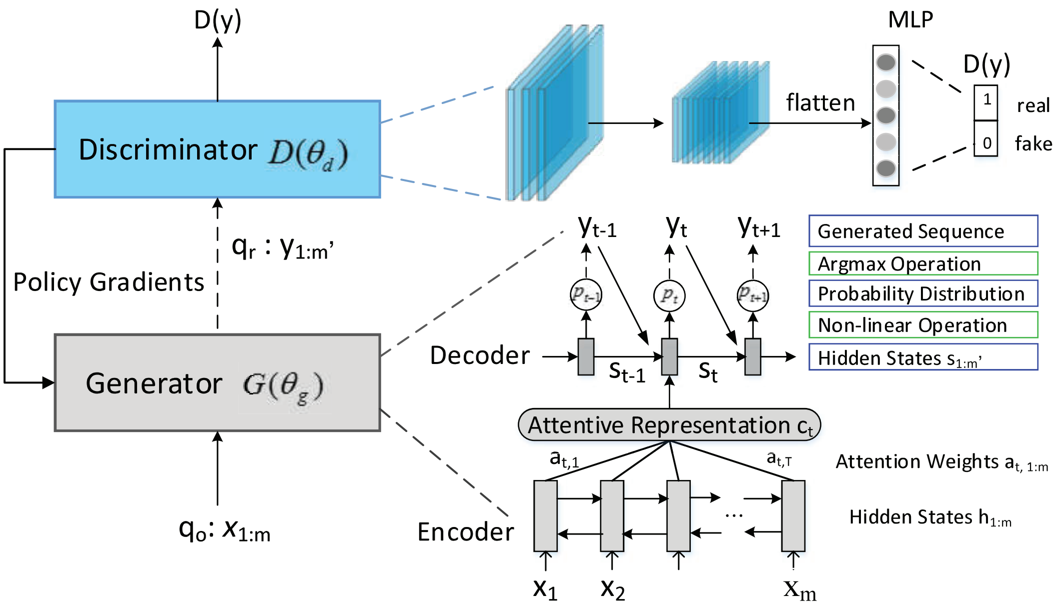


Figure 1: The question generating architecture. D is a regular CNN model, its inputs include generated question q_r and the original question q_o (not show in the picture). G employs the encoder-decoder architecture, but it is different from monolingual machine translation or autoencoder, since there is no target sequence in training phase. Instead, the generating direction is guided by the policy gradients from D.

Policy Gradients Training Though $p(y_t|y_{<t}, x)$ is differentiable w.r.t. generator parameters θ_g, after the *argmax* operation, y_t is no longer differentiable w.r.t. θ_g. The gradients w.r.t. θ_g in Eq. 2 would be zero and θ_g will not be updated. To address this problem, we leverage the policy gradients (Williams 1992; Sutton et al. 1999) that are commonly used in reinforcement learning tasks such as games. Here, we regard G as a policy network whose expected reward(loss) is:

$$L = \mathbb{E}_{y\sim P_{\theta_g}}\left[R\left(y|x\right)\right] = \sum\nolimits_{y\in P_{\theta_g}} Q\left(y\right)G\left(y|x\right) \qquad (8)$$

where $Q(y)$ is action-value function, i.e., the reward for generating sequence y. $G(y|x)$ is the estimated probability of generating sequence y condictioned on x by the policy network. Since G wants to fool D, i.e., maximize $D(y)$. We convert the problem into minimizing L and define the reward $Q\left(y\right) = \log\left(1 - D\left(y\right)\right)$. Noticing y is independent of θ_g, so $\nabla_{\theta_g} Q(y) = \nabla_{\theta_g}\log(1 - D(y)) = 0$. We have:

$$\begin{aligned}\nabla_{\theta_g} L &= \sum\nolimits_{y\in P_{\theta_g}} \left[\nabla_{\theta_g} Q\left(y\right) \cdot G(y|x) + Q(y) \cdot \nabla_{\theta_g} G(y|x)\right] \\ &= \sum\nolimits_{y\in P_{\theta_g}} Q(y) \cdot \nabla_{\theta_g} G(y|x) \\ &= \sum\nolimits_{y\in P_{\theta_g}} Q(y) \cdot G(y|x) \cdot \nabla_{\theta_g} \log G(y|x) \\ &= \mathbb{E}_{y\sim P_{\theta_g}}\left[\log(1 - D(y)) \cdot \nabla_{\theta_g} \log G(y|x)\right] \qquad (9)\end{aligned}$$

Since G wants to maximize $D(y)$ or minimize L, the generator parameters θ_g are updated by:

$$\theta_g \leftarrow \theta_g - \alpha \nabla_{\theta_g} L \qquad (10)$$

Eq. 10 and Eq. 8 denote how we should change the policy network G's predictions(through its parameters θ_g) to minimize the loss and how its actions (generating proper words of y) affect the reward.

Augmented Mutual Information A potential interpretation for the rewriting question is that it enhances mutual information between QA. In information theory, mutual information between Q and A, $I(Q, A)$, measures the amount of information learned from each other. $I(Q, A)$ is the difference of two entropies:

$$I(Q, A) = H(A) - H(A|Q) = H(Q) - H(Q|A)$$

$H(A)$ denotes information entropy of answer, $H(A|Q)$ denotes uncertainty about A given Q. It has an intuitive interpretation: if Q and A are determinately related($A = f(Q)$), then the uncertainty $H(A|Q) = H(Q|A) = 0$ and $I(Q, A)$ is maximized; if Q and A are independent, then $H(A|Q) = H(A)$ and $H(Q|A) = H(Q)$, $I(Q, A)$ is zero. For a pair of QA, if the question could be inferred from the answer or vice versa, then they get the maximized mutual information. A good matching model should reveal as much mutual information as possible. Given a fixed matching model, if we can generate a new question Q_2, then the uncertainty about A is decreased: $H(A|Q_1, Q_2) < H(A|Q_1)$. Therefore, we have:

$$\begin{aligned}I(Q_1, Q_2, A) &= H(A) - H(A|Q_1, Q_2) \\ &> H(A) - H(A|Q_1) = I(Q_1, A)\end{aligned}$$

Hence, the mutual information is enhanced after we rewrite a new question. Since we measure an answer in the embedding space instead of traditional word-matching perspective, the question q_o might have little intersection with the content-rich answer in high dimensional manifolds. If we only use the original question q_o to perform matching, only limited information about answer can be uncovered. If we reformulate the asker's intent in another expression q_r, the mutual information can be further enhanced on basis of q_o. As a result, more aspects of the answer could be uncovered.

3.2 Matching Model

For each question q_o, the model first generates a rewriting question q_r, then encodes q_o, q_r and answer into hidden states. To merge the mutual information of q_o, q_r and the answer, we blend the question (q_o or q_r) and answer into a fusion matrix derived from bilinear transformation of their hidden states, then use a convolutional feature extractor to extract relevance information and flatten it to a dense vector. We concatenate the dense vectors for q_o and q_r, then feed it to multilayer perceptron to compute a relevance score for each answer.

Encoding and Matching. Due to the ambiguity of language, a word in different context will have different meaning. Therefore, when learning representation of a word in a sentence, its context information must be taken into consideration. Bidirectional recurrent neural networks (Graves and Schmidhuber 2005) have been proven to be effective in utilizing context information. Hence, we use bidirectional GRU (Cho et al. 2014) networks to learn the latent representation of each word. The encoding module has two bidirectional RNNs to learn QA's hidden representations. q_o and q_r share the same encoder BiGRU$_1$ and answers share the same encoder BiGRU$_2$. All words are projected into vectors using Glove[1] embedding. Then question and answer are fed to BiGRU$_1$ and BiGRU$_2$ respectively. The encoder reads sequence in normal and reverse order, the forward and backward hidden states are concatenated in the same way as the question rewriting module. Each hidden state $\boldsymbol{h}_t$ of BiGRU at time step t is computed with its context words as following:

$$\begin{aligned}\overrightarrow{h_t} &= \overrightarrow{GRU}(\overrightarrow{h_{t-1}}, w_t)\\ \overleftarrow{h_t} &= \overleftarrow{GRU}(\overleftarrow{h_{t+1}}, w_t)\end{aligned} \tag{11}$$

where hidden state $h_t \in \mathbb{R}^l$, l denotes the size of state after concatenating the forward output and backward output: $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$. We pack the concatenated states $(h_1, h_2, ...h_m)$ to form a 2D matrix, i.e., for question, the matrix $Q \in \mathbb{R}^{m \times l}$; for answer, the matrix $A \in \mathbb{R}^{n \times l}$, where m, n denote length of question and answer. We merge question and answer into fusion matrix using bilinear transformation of their representations:

$$F = \tanh(QWA^T) \tag{12}$$

where $W \in \mathbb{R}^{l \times l}$ is learnable parameter. Without $tanh$ operation, each element F_{ij} is a weighted multiplication of $Q_{i,:}$ and $A_{j,:}$:

$$F_{ij} = \sum_{p=1}^{l} \sum_{q=1}^{l} Q_{iq} W_{qp} A_{jp} \tag{13}$$

This is a soft alignment of $Q_{i,:}$ and $A_{j,:}$ with learnable coefficients. From Eq. 13, we can see cosine function is a special case of the bilinear transformation. Noticing both $Q_{i,:}$ and $A_{j,:}$ contain the context information, this comparison can learn far more complicated relations between QA than simply squashing question and answer into document vectors.

Since our goal is to compute a relevance score for each answer, we need to summarize the fusion matrix into a scalar. We use a CNN model for that because CNN is well known for its ability to extract important features from target picture(fusion matrix here). Empirically, we choose 2×2 and 3×3 convolutional kernels as feature extractors, respectively corresponding to 2-gram and 3-gram alignments of two sequences. We get feature maps from the convolutional extractor and flatten them to a dense vector d_o which contains mutual information $I(q_o, A)$. For the rewriting question q_r, we can similarly get another dense vector d_r. We concatenate d_o and d_r to form $d = [d_o, d_r]$ and feed d to a multi-layer perceptron for final score. d contains the mutual information $I(q_o, q_r, A)$. For a list of answers, we can compute a list a scores $s_1, s_2, ..., s_M$ (M is the number of answers) following above process.

3.3 Ranking Function

The predicted order $o_1, o_2, ..., o_M$ is derived from the relevance scores $s_1, s_2, ..., s_M$. For example, if $M = 5$ and the ground truth order($l_1, l_2, ..., l_M$) is (2, 5, 1, 3, 4), the predicted scores are (0.3, 0.6, 0.4, 0.2, 0.7), then the predicted order(larger is better) is (2, 4, 3, 1, 5). We need to alter the mutual relations of scores(through model parameters) to make them fit for the ground truth order. To accomplish this objective, we need a listwise ranking loss as the training objective. We use LambdaRank (Burges, Ragno, and Le 2006) to compute the ranking loss, where the probability that answer $\boldsymbol{A}_i$ is permuted ahead of answer $\boldsymbol{A}_j$ is defined as:

$$P_{ij} = \sigma(s_i - s_j) = \frac{1}{1 + e^{-(s_i - s_j)}} \tag{14}$$

Since the sigmoid function σ is monotonic increasing about $s_i - s_j$, which denotes that the larger s_i is to s_j, the more likely that the i-th answer is able to permute ahead of the j-th answer. Loss function for $\boldsymbol{A}_i$ and $\boldsymbol{A}_j$ is defined as:

$$L_{ij} = -\overline{P_{ij}} log P_{ij} - (1 - \overline{P_{ij}}) log(1 - P_{ij}) \tag{15}$$

where $\overline{P_{ij}} = 1$, if $l_i > l_j$ else $\overline{P_{ij}} = 0$. Substitute the P_{ij} with Eq. 14, then the loss for all answer pairs becomes:

$$L = \sum_{1 \le i,j \le M} log(1 + e^{-S_{ij} * (s_i - s_j)}) \tag{16}$$

[1] http://nlp.stanford.edu/projects/glove

where $S_{ij} = sign(l_i - l_j)$. Assuming θ is model parameter, noticing $\frac{\partial L_{ij}}{\partial s_i} = -\frac{\partial L_{ij}}{\partial s_j}$, then the gradient of L is:

$$\frac{\partial L}{\partial \theta} = \sum_{1 \le i,j \le M} \frac{\partial L_{ij}}{\partial s_i}\frac{\partial s_i}{\partial \theta} + \frac{\partial L_{ij}}{\partial s_j}\frac{\partial s_j}{\partial \theta} \tag{17}$$

$$= \sum_{i=1}^{M}\sum_{j=1}^{M} \lambda_{ij}\left(\frac{\partial s_i}{\partial \theta} - \frac{\partial s_j}{\partial \theta}\right) \tag{18}$$

$$= \sum_{i=1}^{M} \lambda_i \frac{\partial s_i}{\partial \theta} \tag{19}$$

where $\lambda_{ij} = \frac{\partial L_{ij}}{\partial s_i} = -S_{ij}/(1 + e^{S_{ij}*(s_i - s_j)})$ and the S_{ij} is the same as Eq. 16. The λ_i is synthesized from other lambdas:

$$\lambda_i = \sum_{j=1}^{M} (\lambda_{ij} - \lambda_{ji}) \tag{20}$$

Noticing in the process of back propagation, only gradients of loss are needed. Based on this observation, we only need to compute the gradients in Eq. 19 without computing the actual loss in Eq. 16 whose computation complexity is $O(M^2)$. This is of great importance because the back propagation is calculationally expensive.

4 Experiments

In this section, we describe our experiments on answer selection tasks and the end-to-end ranking experiment on ZhihuRank data.

4.1 Answer Selection

Datasets and Experimental Settings The answer selection (AS) task is a special case of ranking where answers are binary annotated. The goal of AS task is to rank relevant answers ahead of those irrelevant ones. We conduct AS experiments on SemEval 2016 task 3 subtask A[2] dataset and the WikiQA[3](Yang, Yih, and Meek 2015) dataset. The SemEval QA dataset is released by QCRI (Qatar computing research institute) and is popularly used to evaluate a system's capability in answer selection. WikiQA is another AS dataset whose questions are sampled from Bing query logs, answers are extracted from Wikipedia based on users clicks.

We use Adam optimizer with learning rate 0.001 to update parameters, learning rate will be de decayed if results have no improvement in valid data. The batch size is 20 and word embeddings are initialized using 100 dimensions glove embedding. For the generating module, hidden state size of the decoder RNN is 256. The discriminator is a regular CNN model with convolutional kernel heights being 2 and 3, the width equals word embedding size. For the fusion matching module, both BiGRU$_1$ and BiGRU$_2$ have 512 hidden units and initial states are zero. The batch size is 20. L2 regularization is used to prevent model from overfitting. The feature extractor CNN consists of convolution and pooling layers and a dense layer, with 3 × 3 kernel size and 2 × 2 pooling window size. Dropout is employed to the hidden outputs with probability 40% during training phase.

Results The evaluate metrics for AS task are mean average precision (MAP) and mean reciprocal rank (MRR). From Table 1 and Table 2, it can be seen that rewriting question has improved the ranking performance when compared with models without rewriting question, which verifies the hypothesis that rewriting question provides extra information for the matching. We have also investigated the impact of ranking versus classification. The classification methods categorize each answer singly while ranking methods handle all the answers simultaneously. The ranking methods have achieved better performance than classification methods, which indicates that we should rank the answers once and for all, instead of categorizing each answer one by one.

Table 1: Results on SemEval.

System	MAP	MRR
Tranlation-model(Guzman et al. 2016)	78.20	86.93
Structural-KeLP(Filice et al. 2016)	79.19	86.42
Non-rewrite + Classify	75.86	84.23
Rewrite + Classify	77.62	86.14
Non-rewrite + Rank	78.03	86.53
Rewrite + Rank	**80.36**	**87.64**

Table 2: Results on WikiQA.

System	MAP	MRR
CNN-Cnt(Yang, Yih, and Meek 2015)	65.20	66.52
Attention-based CNN(Yin et al. 2016)	69.21	71.08
NoiseContrastive(Rao and He 2016)	70.10	71.80
KeyValue Network(Miller et al. 2016)	70.69	72.65
Pairwise Interaction(He and Lin 2016)	70.90	72.34
Attentive Pool(Yin and Schutze 2017)	71.24	72.37
Non-rewrite + Classify	69.92	70.18
Rewrite + Classify	71.83	72.56
Non-rewrite + Rank	72.16	73.22
Rewrite + Rank	**73.48**	**74.30**

4.2 Multi-level Answers Ranking

Datasets For the multi-level answers ranking task, we built a new dataset, ZhihuRank, collected from Zhihu[4], a professional Chinese QA community that has millions of active users daily. In real scenario, it is unreasonable for answers to have only two kinds of relation(relevant or irrelevant) to question. Different from answers selection datasets where answers are binary annotated, answers in ZhihuRank have multi-grade relevance to question. In ZhihuRank, answers are five-level annotated according to their thumb-up numbers generated by user clicks. In total, there are tens of

[2] http://alt.qcri.org/semeval2016/task3

[3] http://aka.ms/WikiQA

[4] https://www.zhihu.com

millions of questions and around forty million answers in Zhihu. We make ZhihuRank public available[5] for research community. We select 20,000 "useable" questions and their answers from the data warehouse of Zhihu to train and test our model.

Baseline Systems We have investigated different matching models to fully demonstrate the performance of question rewriting.

ARC (Hu et al. 2014): includes ARC-I and ARC-II. ARC-I learns representations of two sequences using CNNs. ARC-II focuses on learning hierarchical patterns from interaction space of two sequences and computes similarity using MLP.

Attentive LSTM (Tan et al. 2016): learns question and answer representations using LSTM with attention mechanism(Bahdanau, Cho, and Bengio 2015).

ABCNN (Yin et al. 2016): This approach introduces attention mechanism into CNNs. It learns sentence vectors using CNNs and utilizes mutual information between sentence pair.

Skip-Thought Vectors (Kiros et al. 2015): This is an extension of word2vec that extends skip-gram model to sentence level and can learn sentence vector.

Compare-Aggregate Model (Wang and Jiang 2017): This method uses various comparison functions to match two sequences and has achieved promising performance on sequences matching tasks.

Evaluation Metrics Since answers in ZhihuRank are multi-level annotated, we need to evaluate the ranking performance using IR metrics. We evaluate the ranking results using normalized discounted cumulative gain (NDCG) and expected reciprocal rank (ERR) metrics, which are designed specially for graded relevance and both are commonly used in information retrieval. $NDCG$ is defined as $NDCG = DCG/iDCG$, where DCG is:

$$DCG = \sum_{i=1}^{M} \frac{2^{o_i} - 1}{log(1+i)} \tag{21}$$

$o_{1,..M}$ is the predicted order. $iDCG$ is the ideal DCG computed from ground truth order$(l_1, l_2, ..., l_M)$. In Eq. 21, DCG assumes document at position i is independent of previous documents. In fact, studies (Clarke et al. 2008) have shown the likelihood a user looks through the i-th document depends on how satisfied the user was with previous observed documents. Hence, a ranking metric must be capable of utilizing relations of the whole list. $NDCG$ doesn't take this into consideration while ERR does. Therefore, we also report ERR results in this work. ERR is defined as:

$$ERR = \sum_{r=1}^{M} \frac{R_r}{r} \prod_{i=1}^{r-1} (1 - R_i), R_i = \frac{2^{o_i} - 1}{2^{o_m}} \tag{22}$$

where o_m is the maximum label value(o_m is 4 here). From Eq. 22, we can see that the gain in position r is affected by its previous documents: if previous documents are high-quality(with large o_i), then the gain at position r would be decreased.

Table 3: Results on ZhihuRank.

System	NDCG	ERR
Random	0.4168	0.3027
ARC-I	0.6435	0.5643
ARC-II	0.6847	0.5876
Skip-Thoughts	0.6912	0.6007
Attentive LSTM	0.7135	0.6112
ABCNN	0.7246	0.6198
Compare-Aggregate	0.7475	0.6316
Non-rewrite+Classify	0.7028	0.6031
Rewrite+Classify	0.7451	0.6277
Non-rewrite+Rank	0.7357	0.6345
Rewrite+Rank	**0.7723**	**0.6502**

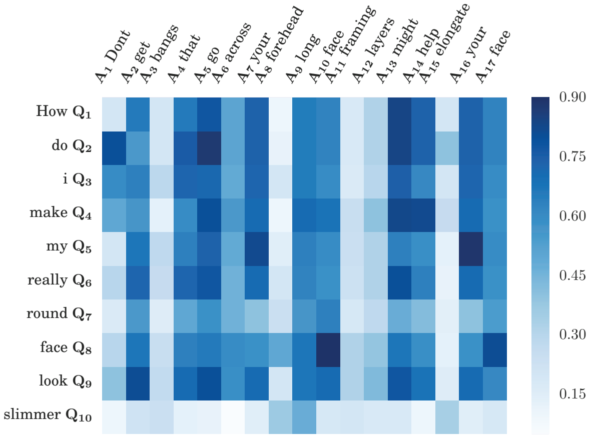


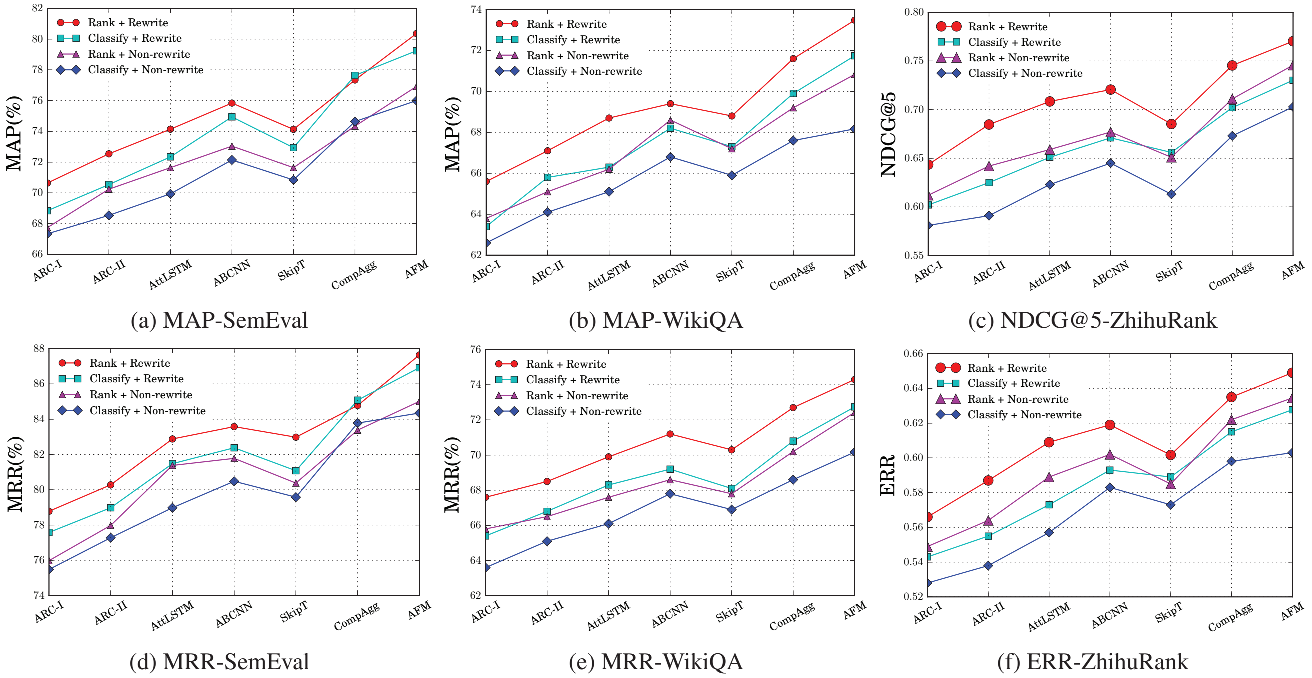
Figure 2: Visualization of the fusion matrix. The heatmap represents a soft alignment between question(left) and its best answer. Deeper color means a better soft alignment.

Results The results are shown in Table 3. For both classification and ranking methods, rewriting question improves test NDCG and ERR compared with models that use only original question. This agrees with our intuition that rewriting question provides extra information and explains original question in another expression, thus matching question and answer more comprehensively. On the other hand, performance of ranking is better than classification. This can be explained that listwise ranking methods are capable of utilizing relationships among different answers and evaluating their relevance to question from global scope, especially when answers have multi-level relevance with question. For classification methods, however, answers are independent from each other and are matched with question singly.

4.3 Visualization

To better understand the matching process, we visualize the fusion matrix of a question and its gold answer. Fig. 2 shows the heatmap of the fusion matrix, which has displayed the matching degrees of different word alignments between QA. From the heatmap, similar word pair obtains a deeper color,

[5] https://goo.gl/zFhS2i



(a) MAP-SemEval

(b) MAP-WikiQA

(c) NDCG@5-ZhihuRank

(d) MRR-SemEval

(e) MRR-WikiQA

(f) ERR-ZhihuRank

Figure 3: Comparisons of rewrite versus non-rewrite, ranking versus classification on SemEval, WikiQA and ZhihuRank datasets.

which means better alignment of words would acquire more attention. It is worth noting that Q_i contains context information around the i-th word, so is A_i. For example, the word "face" appears two times in the answer, but the inner-sentence "face" A_{10} obtains more attention than the tail "face" A_{17}, because the hidden representation learned from BiGRU has summarized both preceding words and succeeding words information with a focus on current word. From the heatmap, we also observe that some columns and rows have very shallow colors, the reason may be that those words appear less frequent than other words and have not been trained sufficiently.

4.4 The Impact of Rewriting and Ranking

We have compared rewriting versus non-rewriting, ranking versus classification to understand their impact on performance. Fig. 3 shows the comparison results on SemEval, WikiQA and ZhihuRank.

Firstly, we observe that models with rewriting question outperform consistently models with only the original question, when other parts keep the same. It corroborates our hypothesis that rewriting question does produce positive impact for the matching model. In other words, the extra expression implicitly enhanced information channels between question and answer, thus more interactions can be captured and more aspects of answer can be revealed.

Secondly, we also notice that ranking loss function performs better than classification loss function(cross entropy as the loss). Different from sentiment analysis where the model only needs to categorize each text independently, for ranking task, there exists partial order relationships among different answer pairs. It would be better to rank the candidate answers as a whole instead of categorizing each answer singly. It is worth noting that the goal is to permutate good answers ahead of bad ones. A system is effective as long as it computes a higher score for gold answer than those bad ones, even if the gold answer obtains a low score. It is important for a model to correctly catch the partial order relations of answers while the actual score value is unimportant. Therefore, a comparing-based loss function such as LambdaRank would be more appropriate than classification loss.

5 Conclusion

In this work, we present that text matching performance can be improved by enhancing mutual information of two sequences. We proposed to generate new text using GAN framework and address the non-differentiability of discrete text by policy gradients. To evaluate the method, we have constructed a multi-level dataset collected from Zhihu. Substantial contrastive experiments have been conducted to investigate various influence factors, including different matching models, ranking versus classification, rewriting versus non-rewriting. We have also visualized the matching process in heatmap. Experimental results have demonstrated that the rewriting question is capable of enhancing mutual information and improving the matching degree. Considering the universal application of text matching, the reformulating mechanism proposed in this work is also promising for many other text matching scenarios, such as paraphrase identification, textual entailment, etc.

Acknowledgments

This work was partially supported by the State Key Laboratory of Software Development Environment of China (No.SKLSDE-2017ZX-16) and the National Natural Science Foundation of China (No. 61332018).

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Bogdanova, D.; Foster, J.; Dzendzik, D.; and Liu, Q. 2017. If you cant beat them join them: Handcrafted features complement neural nets for non-factoid answer reranking. In *EACL*.

Burges, C. J.; Ragno, R.; and Le, Q. 2006. Learning to rank with non-smooth cost functions. In *NIPS*.

Cho, K.; van Merrienboer, B.; aglar Gülehre; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Clarke, C. L. A.; Kolla, M.; Cormack, G. V.; Vechtomova, O.; Ashkan, A.; Büttcher, S.; and MacKinnon, I. 2008. Novelty and diversity in information retrieval evaluation. In *SIGIR*.

Filice, S.; Croce, D.; Moschitti, A.; and Basili, R. 2016. Kelp at semeval-2016 task 3: Learning semantic relations between questions and answers. In *International Workshop on Semantic Evaluation*.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. C.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*.

Graves, A., and Schmidhuber, J. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18:602–610.

Guzman, F.; Marquez, L.; ; and Nakov, P. 2016. Machine translation evaluation meets community question answering. In *ACL*.

He, H., and Lin, J. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Conference of the NNACL: HLT*.

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

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

Jansen, P.; Surdeanu, M.; and Clark, P. 2014. Discourse complements lexical semantics for non-factoid answer reranking. In *ACL*.

Kiros, J. R.; Zhu, Y.; Salakhutdinov, R.; Zemel, R. S.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Skip-thought vectors. In *NIPS*.

Miller, A. H.; Fisch, A.; Dodge, J.; Karimi, A.-H.; Bordes, A.; and Weston, J. 2016. Key-value memory networks for directly reading documents. In *EMNLP*.

Rao, J., and He, H. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *CIKM*.

Reed, S. E.; Akata, Z.; Yan, X.; Logeswaran, L.; Schiele, B.; and Lee, H. 2016. Generative adversarial text to image synthesis. In *ICML*.

Schuster, M., and Paliwal, K. K. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

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

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.

Tan, M.; dos Santos, C. N.; Xiang, B.; and Zhou, B. 2016. Improved representation learning for question answer matching. In *ACL*.

Tymoshenko, K., and Moschitti, A. 2015. Assessing the impact of syntactic and semantic structures for answer passages reranking. In *CIKM*.

Wang, S., and Jiang, J. 2017. A compare-aggregate model for matching text sequences. In *ICLR*.

Wang, D., and Nyberg, E. 2015. A long short term memory model for answer sentence selection in question answering. In *ACL and the 7th IJC-NLP*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8:229–256.

Yang, L.; Ai, Q.; Guo, J.; and Croft, W. B. 2016. anmm: Ranking short answer texts with attention-based neural matching model. In *CIKM*.

Yang, Y.; Yih, W.; and Meek, C. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*.

Yin, W., and Schutze, H. 2017. Task-specific attentive pooling of phrase alignments contributes to sentence matching. In *EACL*.

Yin, W.; Schütze, H.; Xiang, B.; and Zhou, B. 2016. ABCNN: attention-based convolutional neural network for modeling sentence pairs. *TACL* 4:259–272.

Yu, L.; Hermann, K. M.; Blunsom, P.; and Pulman, S. 2014. Deep learning for answer sentence selection. In *NIPS*.

Yu, L.; Zhang, W.; Wang, J.; and Yu, Y. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*.

Zhang, H.; Xu, T.; Li, H.; Zhang, S.; Huang, X.; Wang, X.; and Metaxas, D. N. 2016. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *CoRR* abs/1612.03242.