Bao XQ, Wu YF. A tensor neural network with layerwise pretraining: Towards effective answer retrieval. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 31(6): 1151–1160 Nov. 2016. DOI 10.1007/s11390-016-1689-4

# A Tensor Neural Network with Layerwise Pretraining: Towards Effective Answer Retrieval

Xin-Qi Bao and Yun-Fang Wu*

*Key Laboratory of Computational Linguistics, Peking University, Beijing 100871, China*

E-mail: yikusitian1990@163.com; wuyf@pku.edu.cn

Received July 29, 2015; revised May 27, 2016.

**Abstract** In this paper we address the answer retrieval problem in community-based question answering. To fully capture the interactions between question-answer pairs, we propose an original tensor neural network to model the relevance between them. The question and candidate answers are separately embedded into different latent semantic spaces, and a 3-way tensor is then utilized to model the interactions between latent semantics. To initialize the network layers properly, we propose a novel algorithm called denoising tensor autoencoder (DTAE), and then implement a layerwise pretraining strategy using denoising autoencoders (DAE) on word embedding layers and DTAE on the tensor layer. The experimental results show that our tensor neural network outperforms various baselines with other competitive neural network methods, and our pretraining DTAE strategy improves the system's performance and robustness.

**Keywords** artificial intelligence, language parsing and understanding, machine learning

## 1 Introduction

The answer retrieval problem in community-based question answering (cQA) attracts considerable attention in recent years. Traditional question answering systems, driven by evaluations such as the Text Retrieval Conference (TREC), generally aim to retrieve short and fact-based answers. But questions from cQA services tend to be more subjective and complex, and are often answered in a causal style, including both fact descriptions and subjective opinions, which make the task more challenging. Fig.1 shows a typical case from a travelling forum. The user consults "how about" the hotel's environment, and the answerer mentions various aspects of the hotel according to his/her own experiences.

In this paper, we focus on the answer retrieval problem by modeling the relevance between the question and the answer. Given an input question, we rank the answers from a candidate set through calculating relevance scores between the question and candidate answers.

Q: 请问桂苑宾馆的环境怎么样？

Q: How about the environment of GuiYuan Hotel?

A: 在校园里，安静，也比较安全。硬件还算不错，有空调，服务也可以，就是浴室地板太滑。早上的自助餐我觉得也不错。

A: The hotel locates in campus and thus is quiet and safe. The infrastructure is good and you can use air-conditioning. The service is good but you should take care of the wet floor. I think the buffet there is delicious.

Fig.1. Example of question-answer pair.

Traditional methods based on surface word features such as cosine similarity suffer from the lexical gap problem, which arises from alternative ways of conveying the same information by using synonymy or paraphrasing. The same meaning can be expressed with various representations, and thus to simply match surface words will encounter serious data sparsity problem.

Recently, neural network methods have been used to solve this problem by mapping surface contents into some latent semantic spaces; thus texts talking about similar topics with different surface representations can be mapped to similar points. Moreover, the deep neural network methods have a better representative ability

Regular Paper

This work is supported by the National High Technology Research and Development 863 Program of China under Grant No. 2015AA015403, the National Natural Science Foundation of China under Grant Nos. 61371129 and 61572245, and the Key Program of Social Science Foundation of China under Grant No. 12&ZD227.

*Corresponding Author

than traditional approaches with less human feature engineering. Such studies include the work of Wang *et al.*[1], Hu *et al.*[2] and Lu and Li[3]. Wang *et al.*[1] mapped the question and the answer into a joint latent space and measured the relevance score via some distance metrics or inner product in this latent space. Lu and Li[3] first grouped the word matches into hierarchical patches induced by a topic model, and then used logistic regression to calculate relevance scores.

Nevertheless, it remains a key topic to calculate the relevance score of question-answer pairs. As shown in Fig.1, the relation between the question and the answer is complex, and the length difference is large. Although the question focuses on "environment", the answer gives information concerning various aspects such as "food" and "safety". Therefore the joint representation for question-answer pair in previous work may be insufficient, and distance metrics or inner product used in previous work cannot fully capture the interactions between the question and the answer.

In this paper, we propose an original tensor neural network architecture, where the question and the answer are mapped to two different latent semantic spaces respectively, and a 3-way tensor is employed to model the interaction between their latent semantics. The final relevance score is calculated by logistic regression on the top layer. The network is trained with a max-margin criterion. Moreover, we propose a novel denoising tensor autoencoder (DTAE) algorithm to initialize the tensor layer, and employ a layerwise pretraining procedure on our proposed network. We conduct extensive experiments with different baselines, including cosine similarity, KL divergence, the inner product neural network, and the DEEPMATCH network proposed by Lu and Li[3]. The experimental results show that our tensor neural network outperforms previous work, and our DTAE strategy is remarkably effective.

The main contributions of our work are summarized as follows.

• We propose a novel tensor neural network to model the question-answer relevance, towards effective answer retrieval in cQA services.

• We propose an innovative unsupervised algorithm DTAE to initialize the tensor layer, and employ a layerwise pretraining strategy in our tensor neural network. This method is unsupervised and task-independent, and thus it is potentially useful for a wide range of tensor-based applications.

The rest parts of this paper are organized as follows. Section 2 reviews related work and Section 3 elaborates our network architecture. Section 4 proposes the layerwise pretraining procedure with DTAE. Experimental results are reported in Section 5, and Section 6 concludes the paper.

## 2 Related Work

Extensive studies on deep learning have been done by Hinton *et al.*[4] and Salakhutdinov and Hinton[5], who initially proposed the deep belief nets (DBN).

There are various question answering researches based on deep neural networks. Hu *et al.*[2] used DBN to learn joint representations for textual features and non-textual features, and a linear classification layer on these features was used to predict high-quality answers. Wang *et al.*[1] applied DBN to model the relevance of question-answer pairs in social communities, by calculating the distance in the latent semantic space produced by DBN. Iyyer *et al.*[6] trained a recursive neural network (RNN) based on the dependency tree for factoid question answering, by mapping the question into latent space where it is near to the representation of the entity answer. Bordes *et al.*[7] learnt the joint embedding of words and knowledge base constituents, and they scored questions against candidate answers via inner product.

The key point of the above studies is to obtain the joint distributed representations of the question and the answer in a low-dimensional space. The assumption is that the question and the answer share the same distributed semantics. However, in some cases, it may not make sense since the content of an answer need not strictly correspond to its question in non-factoid community question answering.

A different idea is to extract matching features of question-answer pairs, and employ regression model instead of calculating distance or inner product. Lu and Li[3] proposed a DEEPMATCH architecture where patches are extracted to encode the low-level lexical interaction between the question and the answer, and a multilayer regression model calculates the matching score with the extracted patches. Latent Dirichlet Allocation (LDA) (proposed by Blei *et al.*[8]) is employed to impose a prior sparse structure to the network. But extra hyperparameters are induced by the topic model, which may lead to the uncertainty in system performance. Yih *et al.*[9] proposed an enhanced lexical semantic model for answer selection task. All possible word pairs from both sides are considered as features and fed into an SVM classifier. However, these studies directly extract matching features on the surface

lexical level and thus cannot take advantage of latent semantics of the whole question and answer.

Tensor-based methods can model the multiple interactions of data efficiently, and they are applied to various NLP tasks such as word semantic representation (by Zhang *et al.*[10]), dependency parsing (by Lei *et al.*[11]), word segmentation (by Pei *et al.*[12]), and relation extraction (by Chang *et al.*[13]). In the question answering fields, Yan and Zhou[14] used tensor factorization approach to deal with answerer recommendation. Qiu *et al.*[15] proposed latent semantic tensor indexing (LSTI) method for similar question retrieval. Zhou *et al.*[16] used long short term memory to solve the answer selection task, which aims to identify answer quality from answer sequences rather than to choose a right answer from candidate answers.

A more recent study is done by Qiu and Huang[17]. Both the work of Qiu and Huang[17] and our work employ a tensor network to model question-answer (QA) relevance. However in this paper we propose a novel DTAE strategy to pretrain the tensor layer effectively, which further improves the performance and can be applied to a wide range of tensor-based applications.

Another relevant topic is unsupervised pretraining methods. Pre-training word embeddings on large unlabeled data and using the obtained embeddings to initialize the word layer instead of random initialization can improve the performance (by Mansur *et al.*[18]). Mikolov *et al.*[19] showed that pre-trained embeddings can capture interesting semantic and syntactic information. The ways to learn embeddings on unlabeled data include the one proposed by Mansur *et al.*[18], which learns the embeddings based on neural language model. Mikolov *et al.*[19] proposed a fast skip-gram model Word2Vec, which tries to maximize the classification of a word based on another word in the same sentence. Stacked autoencoder (by Vincent *et al.*[20]) is also an efficient pretraining model to build deep networks. Socher *et al.*[21] used supervised recursive autoencoders for predicting sentiment distributions. Silberer and Lapata[22] used stacked autoencoder to learn high-level embeddings from textual and visual input.

## 3 Network Architecture

The answer retrieval problem is formulated as follows. We are given an archive containing question-answer pairs $C = \{(q_1, a_1), (q_2, a_2), \ldots, (q_N, a_N)\}$, where $N$ is the archive size, and each $a_i$ is the answer for question $q_i$. $\boldsymbol{q}$ and $\boldsymbol{a}$ are represented with bag-of-words vectors $q[k]$ which equals 1 iff the $k$-th word appears in the question.

The task is to learn a relevance scoring model from the archive $C$. Given a question $\boldsymbol{q}$ and a candidate answer set $A$ that contains the correct answer $\boldsymbol{a}$, the model computes the relevance scores between candidate answers in $A$ and the question $\boldsymbol{q}$, and those top scored candidates are treated as probable answers.

### 3.1 Inner Product Model

As mentioned above, directly matching the surface words from the question and the answer tends to produce poor results due to the lexical gap problem. To address this problem, we employ an inner product network architecture, which can be regarded as our baseline model.

Fig.2 depicts the network architecture, where *Rel* denotes the relevance score and *dot* means the inner product computation. The bag-of-words vectors of the question and the answer are separately mapped to the hidden representations with a nonlinear transformation:

$$\begin{aligned} \boldsymbol{h}_q &= \sigma(\boldsymbol{W}_q \cdot \boldsymbol{q} + \boldsymbol{b}_q), \\ \boldsymbol{h}_a &= \sigma(\boldsymbol{W}_a \cdot \boldsymbol{a} + \boldsymbol{b}_a), \end{aligned} \tag{1}$$

where $\boldsymbol{W}_q$ is an $H \times |V_q|$ matrix, and $\boldsymbol{W}_a$ an $H \times |V_a|$ matrix, where $H$ is the dimension of the hidden layer. $\boldsymbol{W}_q$ and $\boldsymbol{W}_a$ have the same number of rows to ensure the hidden representations lie in the same low dimensional semantic space. $\boldsymbol{b}_q$ and $\boldsymbol{b}_a$ are bias vectors, and $\sigma$ is the sigmoid function.

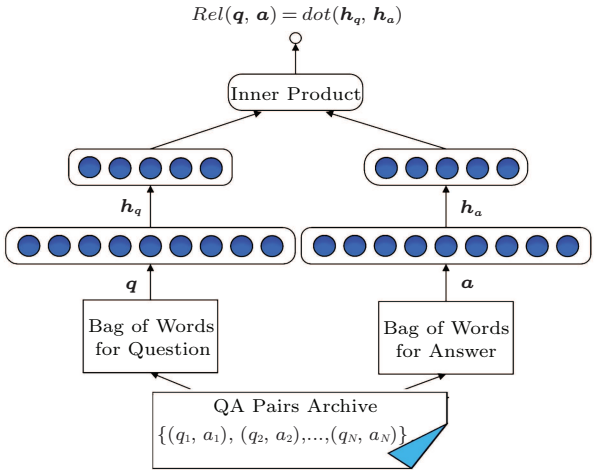



Fig.2. Inner product network architecture.

The relevance score between the question and the answer is computed by the inner product in the latent

semantic space. The sigmoid transformation ensures the score to be in [0, 1] interval.

$$\boldsymbol{s} = \sigma(\boldsymbol{h}_q \cdot \boldsymbol{h}_a).$$

There are two limitations for the inner product model. First, the hidden layers of two sides are constrained to the same size because it assumes uniform latent semantic representation for both the question and the answer. However, in a cQA forum, answers tend to be much longer and contain more information than questions. On average an answer is three times longer than its question according to the statistics of our dataset, for example in Fig.1. Thus it often leads to the information loss of the answer or the information redundancy of the question to embed both the question and the answer into the same latent semantic space.

Second, inner product cannot fully capture the interactions between the question and the answer. It performs some sort of "direct matching" of latent semantic topics which may miss useful information. We use Fig.1 as an example, where the question is asking about the "hotel environment", whereas the answer contains various semantic topics including "environment" "food", "safety", etc. The inner product can only give positive score for the matching of "environment", but misses the relevance of topic pairs between "environment"-"food" and "environment"-"safety", since it just makes a dimension-wise multiplication between the representation vectors of the question and the answer in the latent semantic space.

Generally speaking, the neural network may not induce well-understandable semantics for each dimension of latent space. Thus the above example is just for understanding the model.

We believe that it is hopeful to improve the performance by relaxing the simple matching schema of inner product model. Thus, we propose a tensor-based neural network architecture, which embeds the question and the answer into non-uniform semantic spaces and can explicitly model the interactions between them.

### 3.2 Tensor Model

To model the interactions between the question and the answer in latent spaces, a 3-way tensor is utilized. A 3-way tensor $\boldsymbol{W}$ can be written as $T \times P \times Q$ multi-array, and $\boldsymbol{W}_{ij}^{[k]}$ denotes its $(i, j)$-th element in the $k$-th slice. The 3-way tensor induces a bilinear mapping $\varphi_{\boldsymbol{W}}$: $\mathbb{R}^P \times \mathbb{R}^Q \to \mathbb{R}^T$.

Given $\boldsymbol{x} \in \mathbb{R}^P$, $\boldsymbol{y} \in \mathbb{R}^Q$, the output of tensor transformation $\varphi_W(x, y)$ is a vector $\boldsymbol{z} \in \mathbb{R}^T$:

$$\boldsymbol{z} = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{W}^{[0:T]} \boldsymbol{y},$$

where the $k$-th dimension of $\boldsymbol{z}$ can be calculated as a bilinear form of the $k$-th tensor slice $\boldsymbol{W}^{[k]} \in \mathbb{R}^{P \times Q}$:

$$\boldsymbol{z}_k = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{W}^{[k]} \boldsymbol{y} = \sum_{i=1}^{P} \sum_{j=1}^{Q} \boldsymbol{W}_{ij}^{[k]} \boldsymbol{x}_i \boldsymbol{y}_j.$$

From another point of view, the 3-way tensor transformation can also be seen as a linear transformation $\mathbb{R}^{P \times Q} \to \mathbb{R}^T$ of the outer product vector:

$$\boldsymbol{x} \otimes \boldsymbol{y} = (\boldsymbol{x}_1 \boldsymbol{y}_1, \boldsymbol{x}_1 \boldsymbol{y}_2, ..., \boldsymbol{x}_1 \boldsymbol{y}_Q, ..., \boldsymbol{x}_P \boldsymbol{y}_1, ..., \boldsymbol{x}_P \boldsymbol{y}_Q),$$
$$\boldsymbol{z} = \boldsymbol{W}^{\otimes} \cdot \ (\boldsymbol{x} \otimes \boldsymbol{y}),$$

where $\boldsymbol{W}^{\otimes}$ is a $(PQ) \times T$ matrix with its $k$-th column being just the vector form of the $k$-th tensor slice $\boldsymbol{W}^{[k]}$.

It can be seen that the 3-way tensor works by implicitly representing all possible interactions between $\boldsymbol{x}$ and $\boldsymbol{y}$ as vector formed outer product $\boldsymbol{x} \otimes \boldsymbol{y}$, and mapping the interaction representation into a latent space with linear transformation. This lies in the core of the representative power of our tensor-based model.

Our relevance calculating model is shown in Fig.3. We first embed the question and the answer into the lower dimensional semantic space respectively like in the inner product model, as shown in (1). Note that the sizes of $\boldsymbol{h}_q$ and $\boldsymbol{h}_a$ do not have to be of the same size now.

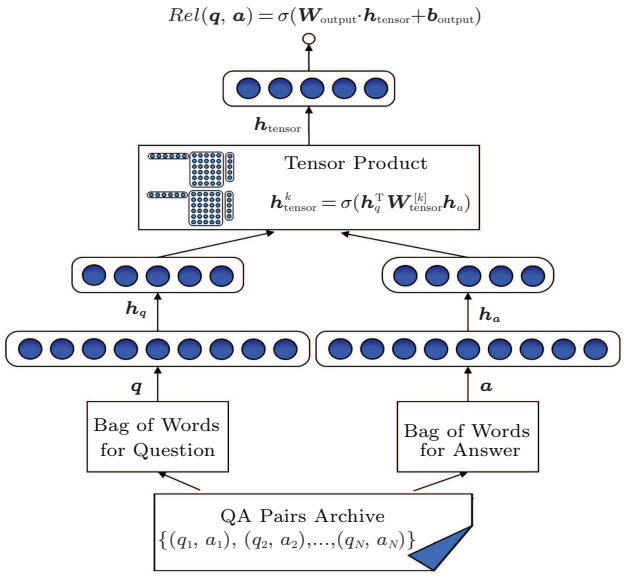



Fig.3. Tensor neural network architecture.

A new hidden layer $\boldsymbol{h}_{\text{tensor}}$ is added by tensor transformation on embeddings $\boldsymbol{h}_q$ and $\boldsymbol{h}_a$:

$$\boldsymbol{h}_{\text{tensor}} = \sigma(\boldsymbol{z}_{\text{tensor}}),$$
$$\boldsymbol{z}^k_{\text{tensor}} = \boldsymbol{h}_q^{\text{T}} \boldsymbol{W}^{[k]}_{\text{tensor}} \boldsymbol{h}_a = \sum_{i=1}^{H_q} \sum_{j=1}^{H_a} \boldsymbol{W}^{[k]}_{\text{tensor}[i,j]} \boldsymbol{h}_{q[i]} \boldsymbol{h}_{a[j]},$$

where $\boldsymbol{W}_{\text{tensor}}$ is a 3-way tensor with size $T \times H_q \times H_a$, where $T$ is the size of tensor layer and $H_q$ and $H_a$ the size of $\boldsymbol{h}_q$ and $\boldsymbol{h}_a$ respectively.

As discussed above, $\boldsymbol{h}_{\text{tensor}}$ is a $T$-dimensional lower dimensional embedding for $\boldsymbol{h}_q \otimes \boldsymbol{h}_a$, which encodes all possible matchings between two latent semantic spaces. Thus $\boldsymbol{h}_{\text{tensor}}$ can be seen as an automatically constructed feature vector, which effectively summarizes the matching degree between the question and the answer.

The final relevance score is calculated by the logistic regression function on the hidden tensor layer $\boldsymbol{h}_{\text{tensor}}$:

$$\boldsymbol{s} = \sigma(\boldsymbol{W}_{\text{output}} \cdot \boldsymbol{h}_{\text{tensor}} + \boldsymbol{b}_{\text{output}}),$$

where $\boldsymbol{W}_{\text{output}}$ is a weight vector with size $T$ and $\boldsymbol{b}_{\text{output}}$ a bias.

Despite the effectiveness of tensor for automatically capturing the matching features for question-answer pairs, computing tensor product is very time-consuming with $O(T \times H_q \times H_a)$ complexity. Low rank technique is employed to decrease the parameter size and accelerate training. The tensor object $\boldsymbol{W}_{\text{tensor}}$ is decomposed into the product of two low-rank subtensors $\boldsymbol{L}_{\text{tensor}}$ and $\boldsymbol{R}_{\text{tensor}}$ with size $T \times H_q \times R$ and $T \times H_a \times R$ respectively:

$$\begin{aligned}\boldsymbol{W}^{[k]}_{\text{tensor}} &= \boldsymbol{L}^{[k]}_{\text{tensor}} \cdot \boldsymbol{R}^{[k]}_{\text{tensor}},\\ \boldsymbol{W}^{[k]}_{\text{tensor}[i,j]} &= \boldsymbol{L}^{[k]}_{\text{tensor}[i]} \cdot \boldsymbol{R}^{[k]}_{\text{tensor}[j]}\\ &= \sum_{r=1}^{R} \boldsymbol{L}^{[k]}_{\text{tensor}[i][r]} \cdot \boldsymbol{R}^{[k]}_{\text{tensor}[j][r]}.\end{aligned}$$

Parameter $R$ controls model complexity at the tensor layer. It is typically set to be far smaller than $H_q$ and $H_a$, and thus the computation complexity can be reduced to $O(T \times \max(H_a, H_q) \times R)$.

### 3.3 Training

We employ a large margin objective for model training. The training is conducted on the triple $(q, a_+, a_-)$, where $a_+$ is the right answer for $q$ while $a_-$ is randomly sampled from the answer set. The idea is that the relevance score between a question and its right answer should be higher than a randomly picked one.

Denoting $\boldsymbol{W}$ as all of the model parameters ($\boldsymbol{W}_q$, $\boldsymbol{b}_q$, $\boldsymbol{W}_a$, $\boldsymbol{b}_a$, $\boldsymbol{W}_{\text{tensor}}$, $\boldsymbol{W}_{\text{output}}$, $\boldsymbol{b}_{\text{output}}$), we minimize the following loss function:

$$J(\boldsymbol{W}) = \frac{1}{N} \sum_{i=1}^{N} e_i(\boldsymbol{W}) + \lambda R(\boldsymbol{W}),$$

where $R(\boldsymbol{W})$ is the L2-norm regularization term for parameter $\boldsymbol{W}$ and $e_i(\boldsymbol{W})$ is the error of the triple $(q_i, a_{i+}, a_{i-})$ given by the large margin form:

$$e_i(\boldsymbol{W}) = \max(0, m + Rel(\boldsymbol{q}_i, \boldsymbol{a}_{i-}) - Rel(\boldsymbol{q}_i, \boldsymbol{a}_{i+})),$$

where $m$ controls the margin of training process.

The objective loss function is optimized with $\boldsymbol{W}$ using stochastic gradient descent (SGD). Because max function is not differentiable, subgradient method is employed:

$$\frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}} = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\partial s(\boldsymbol{q}_i, \boldsymbol{a}_{i+})}{\partial \boldsymbol{W}} - \frac{\partial s(\boldsymbol{q}_i, \boldsymbol{a}_{i-})}{\partial \boldsymbol{W}}\right) + \lambda \boldsymbol{W}.$$

Parameters are initialized via a new layerwise pretraining procedure. We leave the details to the next section.

## 4 Layerwise Pretraining

In our deep tensor neural network architecture, the tensor layer actually preforms feature extraction for matching question-answer pairs. We are curious about whether the tensor product layer can also be properly initialized with some unsupervised method, since the initialization issue of tensor network has not been explored yet.

### 4.1 Autoencoder

Autoencoder is an unsupervised network method to learn latent representations that retain useful features to effectively reconstruct the inputs. It includes two phrases of encoding and decoding.

The encoding phase maps the input vector $\boldsymbol{x}$ to the latent representation $\boldsymbol{y}$ as:

$$\boldsymbol{y} = \sigma(\boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b}).$$

Then the decoding phase aims to reconstruct the input $\boldsymbol{x}$ from the latent representation $\boldsymbol{y}$:

$$\boldsymbol{x}' = \sigma(\boldsymbol{W}^{\text{T}} \cdot \boldsymbol{y} + \boldsymbol{b}').$$

The training objective is to minimize the average reconstruction error measured by cross entropy which can be optimized via gradient descent:

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} rec\_err(\boldsymbol{x}_i, \boldsymbol{x}'_i),$$

$$rec\_err(\boldsymbol{x}, \boldsymbol{x}') = \sum_{k=1}^{P} \boldsymbol{x}_k \times \log(\boldsymbol{x}'_k) + (1 - \boldsymbol{x}_k) \times \log(1 - \boldsymbol{x}'_k),$$

where $\boldsymbol{\theta}$ denotes all parameters $(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{b}')$.

If we corrupt the input $\boldsymbol{x}$ with noises $noise(\boldsymbol{x})$ when calculating $\boldsymbol{y}$, we get the denoising autoencoder, which is more robust to the overfitting problem:

$$\boldsymbol{y} = \sigma(\boldsymbol{W} \cdot noise(\boldsymbol{x}) + \boldsymbol{b}).$$

Moreover, autoencoders can be stacked and then we get stacked (denoising) autoencoders. It is trained layer by layer with the current layer being fed the latent representation of the previous autoencoder as input. Using this unsupervised pre-training, initial parameters are found to approximate a good solution.

### 4.2 Denoising Tensor Autoencoder

We apply the idea of autoencoders to the tensor transformation layer and propose the denoising tensor autoencoder (DTAE) model, as shown in Fig.4. The encoding phase of DTAE is similar to what is discussed in Section 3, which encodes the interaction between two vectors $\boldsymbol{x} \in \mathbb{R}^P$, $\boldsymbol{y} \in \mathbb{R}^Q$ into $\boldsymbol{z} \in \mathbb{R}^T$ by:

$$\boldsymbol{z} = \sigma(\boldsymbol{x}^{\mathrm{T}} \boldsymbol{W}^{[0:T]} \boldsymbol{y}),$$

where $\boldsymbol{W}$ is a $T \times P \times Q$ 3-way tensor, $\boldsymbol{x}$ means the representations of questions and $\boldsymbol{y}$ means answers in our task.

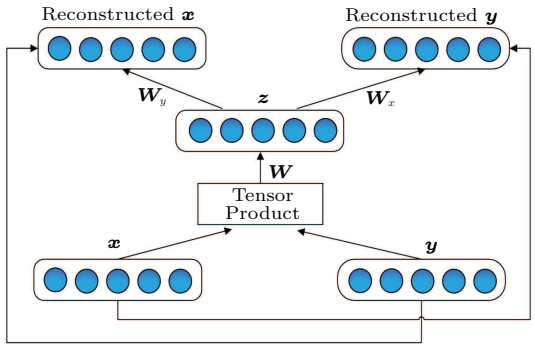



Fig.4. DTAE model.

In the decoding phase, unlike traditional autoencoders, the inputs are not directly reconstructed from $\boldsymbol{z}$. Conversely, it operates as follows.

1) Given the latent representation $\boldsymbol{z}$ and input $\boldsymbol{x}$, we can effectively reconstruct $\boldsymbol{y}$;

2) Given the latent representation $\boldsymbol{z}$ and input $\boldsymbol{y}$, we can effectively reconstruct $\boldsymbol{x}$.

Note that the 3-way tensor $\boldsymbol{W}$ represents a bilinear mapping $\mathbb{R}^P \times \mathbb{R}^Q \to \mathbb{R}^T$, thus fixing either side of the inputs, for example, we fix $\boldsymbol{y} \in \mathbb{R}^Q$, induces a linear mapping $\mathbb{R}^P \to \mathbb{R}^T$ represented by:

$$\boldsymbol{z} = \sigma(\boldsymbol{W}_y \cdot \boldsymbol{x}),$$

where $\boldsymbol{W}_y$ is a $T \times P$ matrix calculated as:

$$\boldsymbol{W}_{y[ij]} = (\boldsymbol{W}^{[0:T]}\boldsymbol{y})_{ij} = \sum_{q=1}^{Q} \boldsymbol{W}^i_{jq} \boldsymbol{y}_q.$$

Given $\boldsymbol{z}$ and $\boldsymbol{y}$, $\boldsymbol{x}$ can be reconstructed as:

$$\boldsymbol{x}' = \sigma(\boldsymbol{W}_y^{\mathrm{T}} \cdot \boldsymbol{z}).$$

Given $\boldsymbol{z}$ and $\boldsymbol{x}$, we can get $\boldsymbol{y}$ likely:

$$\boldsymbol{y}' = \sigma(\boldsymbol{W}_x^{\mathrm{T}} \cdot \boldsymbol{z}).$$

The model is optimized to jointly minimize the reconstruction error ($rec\_err$) for $\boldsymbol{x}$ and $\boldsymbol{y}$ fixing the other side with stochastic gradient descent:

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} rec\_err(\boldsymbol{x}_i, \boldsymbol{x}'_i) + rec\_err(\boldsymbol{y}_i, \boldsymbol{y}'_i).$$

The denoising technique is employed by adding random noises on inputs during training:

$$\boldsymbol{z} = \sigma(noise(\boldsymbol{x})^{\mathrm{T}}, \boldsymbol{W}^{[0:T]} noise(\boldsymbol{y})).$$

As demonstrated in Fig.4, DTAE can be seen as a reversible nonlinear system among latent semantic spaces of the question, the answer, and the matching between them. If the two of three are given, the other one can be effectively estimated, which enforces the output vector $\boldsymbol{z}$ to capture the most useful matching features between $\boldsymbol{x}$ and $\boldsymbol{y}$.

### 4.3 Layerwise Pretraining Procedure

The overall pretraining procedure is described in Fig.5.

In the first layer, the question embedding parameters ($\boldsymbol{W}_q$, $\boldsymbol{b}_q$) and the answer embedding parameters ($\boldsymbol{W}_a$, $\boldsymbol{b}_a$) are independently pretrained with denoising autoencoders to encode $\boldsymbol{q}$ into $\boldsymbol{h}_q$ and $\boldsymbol{a}$ into $\boldsymbol{h}_a$ and minimize the reconstruction error.

Layerwise Pretraining Procedure:

1. Pretrain $(\boldsymbol{W}_q, \boldsymbol{b}_q)$ using DAE:

$$\boldsymbol{W}_q, \boldsymbol{b}_q = \arg\min_{\boldsymbol{W}, \boldsymbol{b}}\left(\frac{1}{N}\sum_{i=1}^{N} rec\_err(\boldsymbol{q}_i, \boldsymbol{q}'_i)\right)$$

2. Pretrain $(\boldsymbol{W}_a, \boldsymbol{b}_a)$ using DAE:

$$\boldsymbol{W}_a, \boldsymbol{b}_a = \arg\min_{\boldsymbol{W}, \boldsymbol{b}}\left(\frac{1}{N}\sum_{i=1}^{N} rec\_err(\boldsymbol{a}_i, \boldsymbol{a}'_i)\right)$$

3. Pretrain $\boldsymbol{W}_{\text{tensor}}$ using DTAE:

$$\boldsymbol{W}_{\text{tensor}} = \arg\min_{\boldsymbol{W}}\left(\frac{1}{N}\sum_{i=1}^{N} rec\_err(\boldsymbol{h}_{q[i]}, \boldsymbol{h}'_{q[i]}) + rec\_err(\boldsymbol{h}_{a[i]}, \boldsymbol{h}'_{a[i]})\right)$$

Fig 5. Layerwise pretraining procedure.

Then, $(\boldsymbol{W}_q, \boldsymbol{W}_a, \boldsymbol{b}_q, \boldsymbol{b}_a)$ are fixed and the embeddings of the question and the answer are fed to the next tensor layer. In the tensor layer, the DTAE algorithm with parameter $\boldsymbol{W}_{\text{tensor}}$ is trained to minimize dual sides' reconstruction errors.

In our experiments, the noise level is set to 0.2, that is, 20% dimensions of inputs are corrupted to zero. The pretraining results are then fed to the tensor neural network architecture as initial weights, which help the supervised learning procedure start from good points.

## 5 Experiment

### 5.1 Experimental Setup

Our dataset comes from XieCheng Traveling forum, which is one of the biggest online travelling forums in China. We crawled 47 000 question-answer pairs from the forum. The data was preprocessed with ICTCLAS Chinese word segmenter[23]. We also trained a CRF-based entity recognizer to label *place* in the data. If the *place* label conflicts between two systems, the label given by the entity recognizer is retained. We removed a small part of question-answer pairs, including repeated pairs, pairs that are too short, and pairs that contain no *place*.

There are 29 506 pairs remained. We pick 23 000 for training, 3 506 for testing and 3 000 for developing. Only content words with POS tag v(erb), n(oun), a(djective) are extracted for bag-of-words representations. The vocabulary size is 8 080 for questions and 22 072 for answers, and the latter is about three times larger than the former, as discussed in Subsection 3.1.

For testing, we retrieve all answers that have the same place entities (NS) with the question as candidate answers, and the candidate answers for each question are more than 20 in our test set.

There are some hyperparameters in the neural network. Table 1 shows the parameter settings.

**Table 1.** Hyperparameter Settings

| Hyperparameter | Value |
|---|---|
| $H_q$ | 150 |
| $H_a$ | 250 |
| $T$ | 200 |
| $R$ | 5 |
| Minibatch size | 50 |
| Learning rate | 0.05 |

### 5.2 Methods

#### 5.2.1 Baselines

We conduct three traditional methods and four neural network methods as our baselines.

*Cosine similarity*: compute the cosine similarity between the question and the answer by using *tf-idf* weight of words.

*KL divergence*: construct unigram language model $M_q$ for the question and $M_a$ for the answer, and then compute their KL-distance.

*Word overlapping*: simply count the number of overlapped words of question-answer pair as the relevance score.

*Q-BoW + A-BoW + Siamese*: as described in Subsection 3.1, use bag-of-words for representations and inner product for relevance calculating. Denoising Autoencoder[20] is used to pretrain the embedding layers.

*Q-CNN + A-CNN + Siamese*: use convolution and pooling layers to extract feature representations and the inner product for relevance calculating. The answers are trimmed to the first 80 words for computational complexity issue.

*DeepMatch*: we implement one version of DeepMatch network proposed by Lu *et al.*[3] This architecture and our network both make regression on matching features of question-answer pairs, but they only considered matching features on the word level by using a topic model to deal with data sparseness problems.

*Q-CNN + A-CNN + MLP*: we also implement the ARC- I network of Hu *et al.*[24] They used a bimodal CNN but with logistic regression layers to calculate matching scores.

#### 5.2.2 Our Methods

We conduct extensive experiments to test our proposed strategies.

*Tensor NN*: use bag-of-words for representations and tensor for question-answer relevance calculating.

*Tensor CNN*: use convolutional NN as representations of questions and answers and tensor for relevance calculating (similar to [17]).

*Tensor NN + DAE*: in tensor NN, we pretrain the word embedding layers by Denoising Autoencoder but randomly initialize the tensor layer.

*Tensor NN + DAE + DTAE*: in tensor NN, our proposed denoising tensor autoencoder (DTAE) model is employed to pretrain both word embedding layers and the tensor layer.

### 5.3 Results

We evaluate the performance of different methods on test data using two metrics: precision@1 (P@1) and mean reciprocal rank (MRR). Table 2 reports the experimental results.

**Table 2.** Experimental Results

| Method | P@1 | MRR |
|---|---|---|
| Cosine | 33.5 | 52.1 |
| KL-Distance | 24.4 | 45.1 |
| Word-Overlapping | 32.4 | 52.0 |
| Q-BoW + A-BoW + Siamese | 48.9 | 60.1 |
| Q-CNN + A-CNN + Siamese | 52.6 | 63.7 |
| DeepMatch | 53.4 | 64.0 |
| Q-CNN + A-CNN + MLP | 55.0 | 65.6 |
| Tensor NN | 52.8 | 63.9 |
| Tensor CNN | 56.1 | 66.8 |
| **Tensor NN +DAE** | **56.5** | **66.4** |
| **Tensor NN + DAE + DTAE** | **58.0** | **67.9** |

The performances of traditional methods, including cosine similarity, KL-divergence and word overlapping are unsatisfying. The network-based methods work considerably better than traditional methods, because they can dive into latent semantic spaces and thus partially overcome the lexical gap problems.

As shown in Table 2, the tensor neural network consistently outperforms the inner product network, both for BoW representations and CNN representations. We think this is because that the tensor network relaxes the joint representation assumption between the question and the answer and thus enables more efficient modelling for question-answer relevance.

Our tensor neural network is also better than DeepMatch, with 4.6% increase in P@1 and 3.9% in MRR. There are two possible reasons. First, DeepMatch employs LDA to keep the neural links sparse, but may cause parameter tuning more difficult. Second, the question-answer interaction is only on the lexical level and is limited to words that belong to the same topic.

### 5.4 Effect of Pretraining

From Table 2, we can also see that the pretraining methods significantly increase the performance and produce better results. In our tensor neural network (Tensor NN + DAE + DTAE), the layerwise pretraining increases 5.2% in P@1 and 4.0% in MRR compared with randomly initialized parameters, where the first layer pretraining with DAE contributes 3.7% in P@1 and 2.5% in MRR while the tensor layer pretraining with DTAE contributes 1.5% in both metrics.

Compared with the more recent work of [17] (Tensor CNN as shown in Table 2), our model obtains a performance gain of 1.9% in P@1 and 1.1% in MRR. Qiu and Huang[17] employed convolutional NN for representations of questions and answers, while we use simple bag-of-words representations. This demonstrates the strong power of our DTAE model.

To further investigate the robustness of the pretraining method, we compare the learning curve of tensor neural network with or without layerwise pretraining as shown in Fig.6.

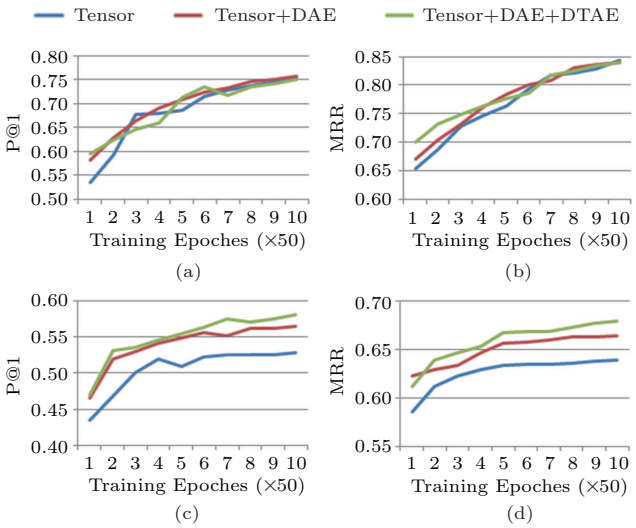



Fig.6. Learning curve on training/testing data. (a) P@1 on training data. (b) MRR on training data. (c) P@1 on testing data. (d) MRR on testing data.

On the training data, the tensor model without pretraining performs almost as well as the model with pretraining, due to the strong representative power

of deep neural network. But on the testing data, the model with pretraining performs consistently better than the one without pretraining. Meanwhile, on both training and testing datasets, the training process starts from a better point for Tensor+DAE and Tensor+DAE+DTAE compared with that without pretraining. We conclude that our layerwise pretraining procedure can benefit training in both the system's performance and robustness.

We further check the effectiveness of denoising tensor autoencoder in an explicit way. We first map the input question and its' corresponding answer into their latent representations $\boldsymbol{h}_q$, $\boldsymbol{h}_a$ respectively, and then get the tensor encoding $\boldsymbol{h}_t$ from them. Now, fixing the latent representation of answer $\boldsymbol{h}_a$, we reconstruct $\boldsymbol{h}_q$ from $\boldsymbol{h}_t$ and $\boldsymbol{h}_a$. Then, from $\boldsymbol{h}_q$ the input bag-of-words representation $\boldsymbol{q}$ is reconstructed via question side DAE. Some of the reconstructed results of question words are shown in Fig.7. We can see that the reconstructed words well capture the meaning of input questions.

### 5.5 Limitations

We have found some question-answer pairs that are not effectively retrieved by the network. If the important words in the question are too rare to be caught by the network, the exact word matching methods tend to perform better. This case may be caused by the following facts. 1) The training data is relatively small to contain enough cases for some words. 2) The training process tends to catch major topics of the data but losses some detailed information. We will try to incorporate the information of exact matching words into the network in the future work.

## 6 Conclusions

In this paper, we proposed a tensor neural network for answer retrieval in cQA services, which employs a 3-way tensor to effectively model the question-answer interaction and relax the joint representation assumption made by previous work. The experimental results showed that our tensor neural network performs much better than various traditional methods and also rivals other competitive neural network methods. What is more, we proposed an original DTAE pretraining strategy to properly initialize the tensor layer, and employed a novel layerwise pretraining procedure for our tensor network, which further significantly improves the performance and robustness.

In future work, we wish to incorporate more structural information and prior knowledge in our tensor neural network, and we also wish to apply our layerwise pretraining strategy to other tensor-based applications such as machine translation.

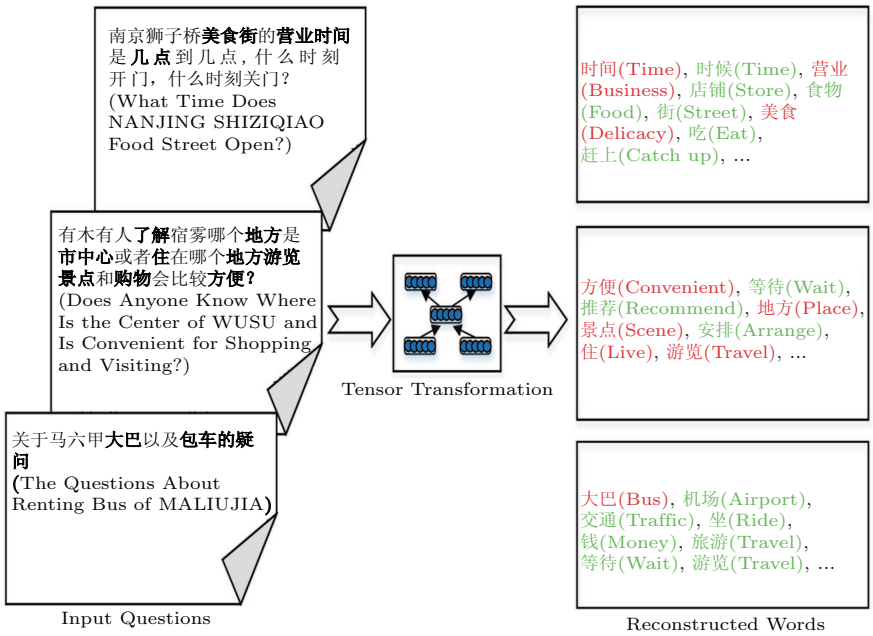



Fig.7. Reconstructed question words.

## References


[1] Wang B, Liu B, Wang X, Sun C, Zhang D. Deep learning approaches to semantic relevance modeling for Chinese question-answer pairs. *ACM Transactions on Asian Language Information Processing* (*TALIP*), 2011, 10(4): Article No. 21.

[2] Hu H, Liu B, Wang B, Liu M, Wang X. Multimodal DBN for predicting high-quality answers in cQA portals. In *Proc. the 51st Annual Meeting of the Association for Computational Linguistics*, August 2013, pp.843-847.

[3] Lu Z, Li H. A deep architecture for matching short texts. In *Proc. the 27th Advances in Neural Information Processing Systems*, December 2013, pp.1367-1375.

[4] Hinton G, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets. *Neural Computation*, 2006, 18(7): 1527-1554.

[5] Salakhutdinov R, Hinton G. Deep Boltzmann machines. In *Proc.the 12th International Conference on Artificial Intelligence and Statistics*, April 2009, pp.448-455.

[6] Iyyer M, Boyd-Graber J L, Boyd J, Claudino L, Socher R, Daumé III H. A neural network for factoid question answering over paragraphs. In *Proc. the 2014 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), October 2014, pp.633-644.

[7] Bordes A, Chopra S, Weston J. Question answering with subgraph embeddings. In *Proc. the 2014 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), October 2014, pp.615-620

[8] Blei D M, Ng A Y, Jordan M I. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 2003, 3: 993-1022.

[9] Yih W T, Chang M W, Meek C, Pastusiak A. Question answering using enhanced lexical semantic models. In *Proc. the 51st Annual Meeting of the Association for Computational Linguistics*, August 2013, pp.1744-1753.

[10] Zhang J, Salwen J, Glass M, Gliozzo A. Word semantic representations using Bayesian probabilistic tensor factorization. In *Proc. the 2014 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), October 2014, pp.1522-1531.

[11] Lei T, Xin Y, Zhang Y, Barzilay R, Jaakkola T. Low-rank tensors for scoring dependency structures. In *Proc. the 52nd Annual Meeting of the Association for Computational Linguistics*, June 2014, pp.1381-1391.

[12] Pei W, Ge T, Chang B. Max margin tensor neural network for Chinese word segmentation. In *Proc. the 52nd Annual Meeting of the Association for Computational Linguistics*, June 2014, pp.293-303.

[13] Chang K W, Yih W T, Yang B, Meek C. Typed tensor decomposition of knowledge bases for relation extraction. In *Proc. the 2014 Conference on Empirical Methods in Natural Language Processing*, October 2014, pp.1568-1579.

[14] Yan Z, Zhou J. A new approach to answerer recommendation in community question answering services. In *Proc. the 34th Advances in Information Retrieval*, April 2012, pp.121-132.

[15] Qiu X, Tian L, Huang X. Latent semantic tensor indexing for community-based question answering. In *Proc. the 51st Annual Meeting of the Association for Computational Linguistics*, August 2013, pp.434-439.

[16] Zhou X, Hu B, Chen Q *et al.* Answer sequence learning with neural networks for answer selection in community question answering. arXiv:1506.06490, 2015. http://arxiv.org/abs/1506.06490, June 2016.

[17] Qiu X, Huang X. Convolutional neural tensor network architecture for community-based question answering. In *Proc. the 24th International Joint Conference on Artificial Intelligence*, July 2015, pp.1305-1311.

[18] Mansur M, Pei W, Chang B. Feature-based neural language model and Chinese word segmentation. In *Proc. the 6th International Joint Conference on Natural Language Processing*, October 2013, pp.1271-1277.

[19] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv:1301.3781, 2013. http://arxiv.org/abs/1301.3781, June 2016.

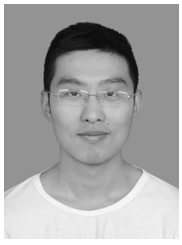
[20] Vincent P, Larochelle H, Bengio Y, Manzagol P A. Extracting and composing robust features with denoising autoencoders. In *Proc. the 25th International Conference on Machine Learning*, June 2008, pp.1096-1103.

[21] Socher R, Pennington J, Huang E H, Ng A Y, Manning C D. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proc. the Conference on Empirical Methods in Natural Language Processing*, July 2011, pp.151-161.

[22] Silberer C, Lapata M. Learning grounded meaning representations with autoencoders. In*Proc. the 52nd Annual Meeting of the Association for Computational Linguistics*, June 2014, pp.721-732.

[23] Zhang H, Yu H, Xiong D, Liu Q. HHmm-based Chinese lexical analyzer ICTCLAS. In *Proc. the 2nd SIGHAN Workshop on Chinese Language Processing*, July 2003, pp.184-187.

[24] Hu B, Lu Z, Li H, Chen Q. Convolutional neural network architectures for matching natural language sentences. In *Proc. Advances in Neural Information Processing Systems*, December 2015, pp.2042-2050.

**Xin-Qi Bao** is currently a Master student at Peking University, Beijing. He received his B.S. degree in electronic engineering and computer science from Peking University, Beijing, in 2013. His main research interests include natural language processing and machine learning.

**Yun-Fang Wu** is currently an associate professor in the Department of Computer Science and Technology, Peking University, Beijing. She received her Ph.D. degree in linguistics from Peking University, Beijing, in 2003. Her main research interests include natural language processing and question answering.