Available online at www.sciencedirect.com

ScienceDirect

Knowledge-Based Systems 00 (2015) 1–15

Knowledge-Based Systems

Learning Semantic Representation with Neural Networks for Community Question Answering Retrieval

Guangyou Zhou[a], Yin Zhou[a], Tingting He[a], Wensheng Wu[b]

[a]*School of Computer, Central China Normal University, Wuhan 430079, China*
[b]*Computer Science Department, University of Southern California, Los Angeles, CA 90089, USA*

Abstract

In community question answering (cQA), users pose queries (or questions) on portals like Yahoo! Answers which can then be answered by other users who are often knowledgeable on the subject. cQA is increasingly popular on the Web, due to its convenience and effectiveness in connecting users with queries and those with answers. In this article, we study the problem of finding previous queries (e.g., posed by other users) which may be similar to new queries, and adapting their answers as the answers to the new queries. A key challenge here is to the bridge the lexical gap between new queries and old answers. For example, "company" in the queries may correspond to "firm" in the answers. To address this challenge, past research has proposed techniques similar to machine translation that "translate" old answers to ones using the words in the new queries. However, a key limitation of these works is that they assume queries and answers are parallel texts, which is hardly true in reality. As a result, the translated or rephrased answers may not look intuitive.

In this article, we propose a novel approach to learn the semantic representation of queries and answers by using a neural network architecture. The learned semantic level features are finally incorporated into a learning to rank framework. We have evaluated our approach using a large-scale data set. Results show that the approach can significantly outperform existing approaches.

Keywords: Community Question Answering, Question Retrieval, Text Mining, Yahoo! Answers

Email address: gyzhou@nlpr.ia.ac.cn (Guangyou Zhou)

1. Introduction

With the development of Web 2.0, community question answering (cQA) services like Yahoo! Answers,[1] Baidu Zhidao[2] and WkiAnswers[3] have attracted both academia and industry great attention [1, 2, 3]. In cQA, anyone can ask and answer questions on any topic, and people seeking information are connected to those who know the answers. As answers are usually explicitly provided by human, they can be helpful in answering real world questions.

One fundamental task for reusing content in cQA is finding the existing answers for queries, as query-answer pairs are the keys to accessing the knowledge in cQA. Many studies have been done along this line [1, 2, 4, 5, 6, 7, 8, 9, 10]. One big challenge for community question answering retrieval is the lexical gap between queries and answers in the archives. Lexical gap means that the queries may contain words that are different from, but related to, the words in the answers. For example, if an user's query contains the word "company" but a target answer contains the word "firm", then there is a lexical gap and the target answer may easily regarded as an irrelevant one. This lexical gap has become a major barricade preventing traditional IR models (e.g., BM25 [11]) from retrieving the target answers in cQA.

To solve the lexical gap problem, previous work in the literature proposed a method to leverage query-answer pairs and learn translation models to improve traditional IR models [1, 2]. The basic assumption is that query-answer pairs are "parallel texts" and relationships of words (or phrases) can be established through word-to-word (or phrase-to-phrase) translation probabilities [1, 2, 8]. Experimental results show that translation models obtain state-of-the-art performance for community question answering retrieval in cQA. However, query-answer pairs are far from "parallel" in practice, there are large number of unaligned words in query-answer pairs than in bilingual pairs, which confuses the word alignment tools [9].

In this paper, we study how to learn query and answer representations for community question answering retrieval. Specially, we head for modeling queries and answers in a more natural way. The model should not only highlight the instinct heterogeneity of queries and answers, but also be flexible enough to take other answers rather than the best answers into account. To this end, we propose a novel supervised approach to automatically learn semantic representations for queries and answers. The underlying assumption is that although questions and answers are heterogeneous in many aspects, they share some equivalences in the semantic level. Thus, we can learn the unified representations for query-answer pairs using an approach based on neural networks. In details, the procedure of using a deep neural network (DNN) to rank a set of answers for a given query is as follows. First, a non-linear projection is performed to map the query-answer pairs to a common semantic space. Then, the relevance of each answer given the query is calculated as the cosine similarity between their vectors in that semantic space. The neural network models are discriminatively trained using the query-answer pairs such that the cosine similarity of the best answer given the query is maximized. Finally, this semantic level feature is incorporated into a learning to rank (LTR) framework, which also includes a rich set of statistical-based features described in [12]. The relative importance of each feature is learned via a SVMRank algorithm [13] that utilizes a large-scale query-answer pairs in cQA archive. Different from the previous semantic models that are learned in an unsupervised fashion [14], our models are optimized directly for queries and the corresponding best answers, and thus give superior performance.

We evaluate the contribution of our semantic level features for community question answering retrieval under two settings: (1) large-scale automatic evaluation over query-answer pairs in cQA archives; (2) manual evaluation of the top retrieved answers for a set of test queries. We compare our approach to a state-of-the-art LTR model that utilizes only statistical-based features. The performance improved significantly in query-answer ranking by both evaluations when the semantic level features are incorporated, demonstrating the potential of our learn scheme for community question answering retrieval.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 describes neural network based approach for question and answer representation. Section 4 presents the learning to rank scoring model. Section 5 presents the experimental results. Finally, we conclude the paper in section 6.

[1]http://answers.yahoo.com/
[2]http://zhidao.baidu.com/
[3]http://wiki.answers.com/

2. Related Work

2.1. Question Retrieval in cQA

In recent years, with the flouring of community question answering (cQA) archives, significant research efforts have been conducted in attempt to improve question retrieval in cQA [1, 2, 5, 4, 6, 7, 8, 15, 16, 17, 9, 3, 18, 19, 20]. Particularly, the effectiveness of methods based on language model is proven. Most cQA researchers focus on leveraging metadata in cQA to improve the performance of the traditional language models for question retrieval [9]. Basically, there are five groups of work. The first group considers leveraging categories of questions. For example, Cao et al. [21] and Cao et al. [7] proposed a language model with leaf category smoothing in which they estimated a new smoothing item for language models from questions under the same category. Cai et al. [22] proposed a topic model incorporated with the category information into the process of discovering the latent topics in the content of questions. Then they combine the semantic similarity based latent topics with the translation-based language model [2] into a unified framework for question retrieval. Zhou et al. [17] proposed a faster and better retrieval model by leveraging category to filter certain amount of irrelevant questions under a wide range of leaf categories. Zhou et al. [3] proposed a novel approach called group non-negative matrix factorization with natural categories for question retrieval. This is achieved by learning the category-specific topics for each category as well as shared topics across all categories via a group non-negative matrix factorization framework. Recently, Zhou et al. [20] proposed to learn continuous word embeddings with metadata of category information within cQA pages for question retrieval. The basic idea is that category information encodes the attributes or properties of words, from which we can group similar words according to their categories. Thus the category information benefits the word embedding learning for question representation.

The second group leverages question-answer pairs to learn various translation models to bridge the lexical gap problem. For example, Jeon et al. [1] proposed a word-based translation model which exploits the semantic similarity among answers of existing questions to learn translation probabilities, which allows them to match semantically similar questions despite lexical gap. Xue et al. [2] proposed a word-based translation language model for question retrieval with a query likelihood model for the answer. Experiments consistently reported that the word-based translation model could yield better performance than the traditional methods (e.g., VSM, BM25 and LM). However, these word-based translation models are considered to be context independent in that they do not take into account any contextual information in modeling word translation probabilities. In order to further improve the word-based translation model with some contextual information, Riezler et al. [23] and Zhou et al. [8] proposed a phrase-based translation model for question and answer retrieval. The phrase-based translation model can capture some contextual information in modeling the translation of phrases as a whole, thus the more accurate translations can better improve the retrieval performance. Furthermore, Singh [15] addressed the lexical gap issues by extending the lexical word-based translation model to incorporate semantic information (entities). However, since it is possible for unimportant words (e.g., non-topical words, common words) to be included in the translation models, a lack of noise control on the models can cause degradation of retrieval performance. Lee et al. [5] investigated a number of empirical methods to eliminate unimportant words in order to construct compact translation models for retrieval purpose. Bernhard and Gurevych [6] proposed to use as a parallel training data set the definitions and glosses provided for the same term by different lexical semantic resources. Besides, Zhou et al. [24] proposed to use of translated words to enrich the question representation, going beyond the words in the original language to represent a question. Zhou et al. [25] proposed to employ statistical machine translation to improve question retrieval and enrich the question representation with the translated words from other languages via matrix factorization. Zhang et al. [19] explored a pivot language translation based approach to derive the paraphrases of key concepts.

The third group applies topic modeling techniques for information retrieval. In recent years, probabilistic topic models have also been introduced to cQA. For example, Cai et al. [22] incorporated question category into the traditional topic model and combined the topic model with a translation-based language model. Ji et al. [26] proposed a question-answer topic model to learn the latent topics aligned across the question-answer pairs to alleviate the lexical gap problem, with the assumption that a question and its paired answer share the same topic distribution. Zhang et al. [9] proposed a supervised question-answer topic modeling approach, which assumes that questions and answers share some common latent topics and are generated in a question language and answer language. Besides, other researchers also applied the topic models for the related tasks in cQA. Guo et al. [27] proposed a generative model to simulate user behaviors in cQA, for both question asking and answering, and then simultaneously obtain topic

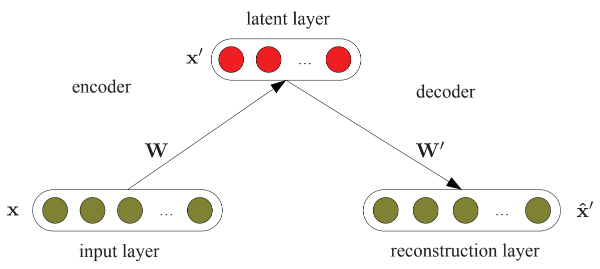

Figure 1. Illustration of an autoencoder.

analysis of questions/answers and users. Then they recommended answer providers for new questions according to discovered topic as well as term-level information of questions and users. Zhou et al. [18] proposed a topic-sensitive probabilistic model by taking into account both the link structure and the topical similarity among users for expert finding.

The fourth group employs the syntactic information for question retrieval. For example, Duan et al. [4] first detected question topic and question focus by using a tree cut method and syntactic parser. They then proposed a new language model to capture the relation between question topic and question focus for question retrieval. After that, Wang et al. [28] proposed a syntactic tree matching model to finding similar questions, and demonstrated that the model is robust against grammatical errors.

The fifth group applies deep learning for question retrieval. Recently, deep learning gains widely interest in natural language processing community. Wang et al. [29] applied the deep belief nets (DBN) to model the relevance of question-answer pairs in cQA, by calculating the distance in the latent semantic space produced by DBN. Hu et al. [30] used a DBN to learn joint representations for textual features and non-textual features, and a linear classification layer on these features is used to predict high quality answers. Lu et al. [31] proposed a DEEPMATCH architecture where patches are extracted to encode low level lexical interaction between question and answer, and a multilayer regression model calculates the matching score from those patches. Mohit et al. [32] trained a recursive neural network (RNN) based on dependency tree for factoid question answering, by mapping the question into latent space. Hu et al. [33] proposed a deep convolutional architecture for matching natural language sentences (e.g., question and answer), which can nicely combine the hierarchical modeling of individual sentences and the patterns of their matching. Qiu and Huang [34] proposed a convolutional neural tensor network architecture to encode the sentences (e.g., question or answer) in semantic space and model their interactions with a tensor layer. This model integrated sentence modeling and semantic matching into a single model, which not only can capture the useful information with convolutional and pooling layers, but also learn the matching metrics between the question and its answer.

However, these works directly extract matching features on the surface lexical feature thus cannot take advantage of deep semantics of the whole question and answers. Besides, we also combine the semantic level features with the statistical-based features into a learn to rank framework.

Besides, some other studies model the semantic relationship between queries and answers with deep linguistic analysis or a learning to rank strategy. Surdeanu et al. [35] and Carmel et al. [10] proposed an approach to rank the answers retrieved by Yahoo! Answers with multiple features. Wang et al. [36] aimed to rank the candidate answers with only word information instead of the combination of different kinds of features. Liu et al. [37] proposed a novel method to recommend related QA documents for knowledge communities of Q&A websites. Song et al. [38] formulated the problematic situations in task processing as knowledge application context (KAC), and proposed to elicit KACs semi-automatically from domain Q&A archives. Romero et al. [39] aimed to provide users with mechanisms to navigate through the domain of knowledge and to facilitate both learning and searching, beyond classic FAQ retrieval algorithms. Moreo et al. [40] proposed a semiautomatic method to reduce the problem of creating templates to that of validate, and possibly modify, a list of proposed templates for question answering systems. Zhou et al. [41] developed a unified framework to leverage the semantic knowledge to enhance the question retrieval in the concept space.

2.2. Auto-encoder

In this subsection, we describe the basic auto-encoder and the denoising auto-encoder, which are related to the proposed approach.

2.2.1. *The Basic Auto-encoder*

An auto-encoder is an unsupervised neural network which is trained to reconstruct a given input vector from its latent representation [42]. It can be seen as a special neural network with three layers: the input layer, the latent layer, and the reconstruction layer (as shown in Figure 1). An auto-encoder contains two parts: (1) The encoder maps an input vector $\mathbf{x}^{(i)} \in \mathbb{R}^m$ to the latent representation $\mathbf{x}'^{(i)} \in \mathbb{R}^{m'}$ via a deterministic mapping f_θ:

$$\mathbf{x}'^{(i)} = f_\theta(\mathbf{x}^{(i)}) = s_e(\mathbf{W}\mathbf{x}^{(i)} + \mathbf{b}) \tag{1}$$

where s_e is the activation function of the encoder, whose input is called the activation of the latent layer, and $\theta = \{\mathbf{W}, \mathbf{b}\}$ is the parameter set with a weight matrix $\mathbf{W} \in \mathbb{R}^{m' \times m}$ and $\mathbf{b} \in \mathbb{R}^{m'}$ is a bias vector. (2) The decoder maps the latent representation $\mathbf{x}'^{(i)}$ back to a reconstruction $\hat{\mathbf{x}}^{(i)} \in \mathbb{R}^m$ via another mapping function $f_{\theta'}$:

$$\hat{\mathbf{x}}^{(i)} = f_{\theta'}(\mathbf{x}'^{(i)}) = s_d(\mathbf{W}'\mathbf{x}'^{(i)} + \mathbf{b}') \tag{2}$$

Similarly, s_d is the activation function of the decoder with parameters $\theta' = \{\mathbf{W}', \mathbf{b}'\}$. The weight matrix $\mathbf{W}'$ of the reverse mapping may optionally be constrained by $\mathbf{W}' = \mathbf{W}^T$, in which case the auto-encoder is said to have tied weights [43, 44]. The training objective is the determination of parameters $\hat{\theta} = \{\mathbf{W}, \mathbf{b}\}$ and $\hat{\theta}' = \{\mathbf{W}', \mathbf{b}'\}$ that minimize the average reconstruction error over a set of input vectors $\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}\}$:

$$\hat{\theta}, \hat{\theta}' = \arg\max_{\theta,\theta'} \frac{1}{n} \sum_{i=1}^{n} L(\mathbf{x}^{(i)}, \hat{\mathbf{x}}^{(i)}) \tag{3}$$

where L is a loss function, such as cross-entropy. Parameters θ and θ' can be optimized by stochastic or mini-batch gradient descent. By minimizing the reconstruction error, we require the latent features should be able to reconstruct the original input as much as possible. In this way, the latent features preserve regularities of the original data.

2.2.2. *The Denoising Auto-encoder*

The procedure of training a deep network using the denoising auto-encoder is similar to the basic auto-encoder. The only difference is how each layer is trained. The training criterion with denoising auto-encoders is to reconstruct a clean "repaired" input from a corrupted, partially destroyed one. This is done by first corrupting the initial input vector $\mathbf{x}^{(i)}$ to get a partially destroyed version $\widetilde{\mathbf{x}}^{(i)}$ [44]. The basic idea is that the learned latent representation is good if the auto-encoder is capable of reconstructing the actual input from its corruption. The advantage of the denoising auto-encoder is that it is much more robustness for the real world data set. The loss function of the reconstruction error for an input vector $\mathbf{x}^{(i)}$ becomes:

$$L(\mathbf{x}^{(i)}, f_{\theta'}(f_\theta(\widetilde{\mathbf{x}}^{(i)}))) \tag{4}$$

Following the literature [44], we consider one possible corrupting process, parameterized by the desired proportion v of "destruction": for each input vector $\mathbf{x}^{(i)}$, a fixed number of vm of components are chosen at random, and their values are forced to 0, while the others are left untouched.

3. Learning Semantic Representation with Neural Networks

In this section, we describe our neural network based query and answer representation model in details. There are two phases in our neural network training process: *pre-training* and *fine-tuning*. In the *pre-training* phase, we build two neural networks with the same structure but different parameters to learn a low-dimensional representation for query part or answer part. Then, in the *fine-tuning* phase, our model directly optimizes the similarity of two low-dimensional representations, so that it highly correlates to our community question answering retrieval task. Finally, we transform the new queries and answers from the retrieval data collection (rather than the training data collection) into the low-dimensional representation with the learned parameters and model the community question answering retrieval in the unified space of queries and answers.

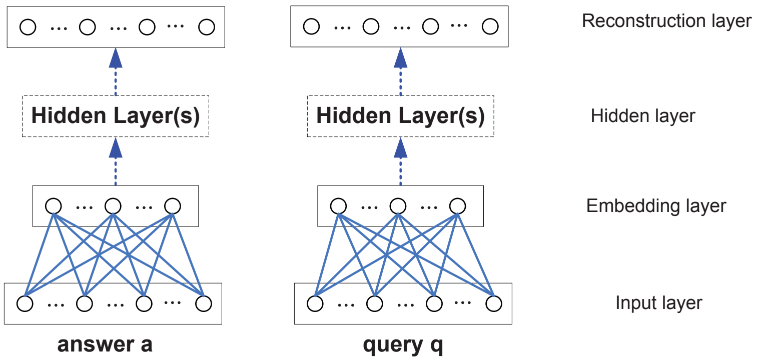

Figure 2. An illustration of the neural network pre-training process based on query-answer pairs.

3.1. Pre-training Using Denoising Auto-encoder

Figure 2 illustrates the high-level framework which describes how to learn semantic representations using query-answer pairs in the pre-training stage, the first layer is the bag-of-words representation of the query and answers, the second layer is the continuous word embedding spaces, the third layer is the latent semantic spaces, and the fourth layer is the reconstruction layer. Given a query-answer pair $< q, a >$, we first transform the query q and the corresponding answer a into the continuous word embedding spaces E_q and E_a, where V is the vocabulary size.

Then, we use neural network structure to transform high-dimensional embedding vectors to low-dimensional dense vectors. We calculate the similarity between the query-answer pair on top of the learned dense vectors. This dense representation should preserve the information from the bag-of-embedded-words input, meanwhile alleviates the data sparseness and lexical gap problems. Therefore, we leverage a specially designed mechanism called auto-encoder to solve this problem. Auto-encoder is one of the basic blocks of deep learning [43]. It can be seen as a special neural network with three layers: the input layer, the encoding layer and the decoding layer. Assuming that the input is a continuous word embedding vector $E_\mathbf{x}$ of a sentence x, an auto-encoder consists of an encoding process $f(E_\mathbf{x})$ and a decoding process $g(f(E_\mathbf{x}))$. The objective of the auto-encoder is to minimize the reconstruction error $\mathcal{L}(g(f(E_\mathbf{x})), E_\mathbf{x})$. Our goal is to learn a low-dimensional vector which can preserve information from the original input vector. Noting that our proposed model uses multiple subtle semantic differences between (query, best answer) and (query, other answers), which is different from the single semantic model presented in [45].

Auto-encoder is a means to learn representations of some input by retaining useful features in the encoding phase which helps to reconstruct the input, whilst discarding useless or noisy ones. One problem with auto-encoder is that it treats all words in the same way, making no difference between function words (e.g., the) and content words (e.g., good). The representation learned by auto-encoders tends to be influenced by the function words, thereby it is not robust. To tackle this problem, Vincent et al. [44] proposed the Denoising Auto-Encoder (DAE), which aims to reconstruct a clean "repaired" input from a corrupted, partially destroyed one. This is done by first corrupting the initial input vector $E_\mathbf{x}$ to get a partially destroyed version $\tilde{E}_\mathbf{x}$. The basic idea is that the learned semantic representation is good if the auto-encoder is capable of reconstructing the actual input from its corruption.

3.1.1. Continuous Word Embedding Representations

We pre-train the word embedding representations with an unsupervised learning method. In this paper, we consider a context-aware predicting model, more specifically, the *Skip-gram* model [46] for learning word embeddings, since it is much more efficient as well as memory-saving than other approaches. *Skip-gram* is recently proposed for learning word representations using a neural network model, whose underlying idea is that similar words should have similar context. After learning via gradient ascent, we can get the resulting matrix of word embedding $L \in \mathbb{R}^{d\times|V|}$, where d is the dimension of each word embedding.

Assume that we are given a sentence (e.g., a query or answer) as an ordered list of m words. Each word has an associated vocabulary index k incorporated into the embedding matrix which we use to retrieve the word's vector representation. Mathematically, this look-up operation can be seen as a simple projection layer where we use a binary

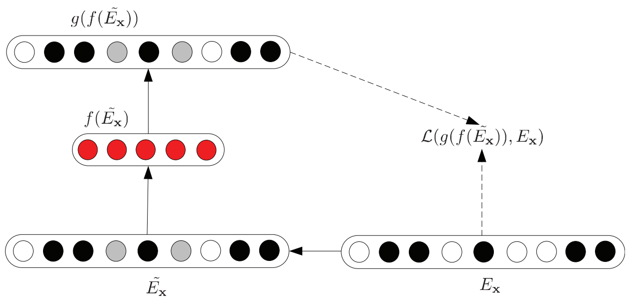
$g(f(\tilde{E}_{\mathbf{x}}))$
$f(\tilde{E}_{\mathbf{x}})$
$\mathcal{L}(g(f(\tilde{E}_{\mathbf{x}})), E_{\mathbf{x}})$
$\tilde{E}_{\mathbf{x}}$
$E_{\mathbf{x}}$

Figure 3. A denoising auto-encoder with a continuous word embedding vector input.

vector $\mathbf{b}'$ which is zero in all position except at the kth index,

$$\mathbf{x}_i = L\mathbf{b}'_k \in \mathbb{R}^d \tag{5}$$

In our task, for each query (or answer) $x = (x_1, x_2, \cdots, x_m)$ with m words, the word embedding of x_i is $\mathbf{x}_i$. Thus, the continuous word embedding vector $E_{\mathbf{x}}$ can be represented by the concatenation of each of the word embedding vector, that is $E_{\mathbf{x}} = (\mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \cdots \mathbf{x}_m) \in \mathbb{R}^{md}$. Once we obtain the continuous word embedding vector $E_{\mathbf{x}}$ shown in Figure 3, we consider one possible corrupting process with DAE, parameterized by the desired proportion μ of "destruction": for each input vector $E_{\mathbf{x}}$, a fixed number of μmd of components are chosen at random, and their values are forced to 0, while the others are left untouched.

3.1.2. Encoding Process

In our case, the encoding process transforms the corrupted input $\tilde{E}_{\mathbf{x}}$ into $f(\tilde{E}_{\mathbf{x}})$ with two layers: a embedding layer connected with a hidden layer. Assuming that the dimension of $f(\tilde{E}_{\mathbf{x}})$ is d_1, the linear layer forms a $d_1 \times md$ matrix $\mathbf{W}$ which projects the md dimensional vector to a d_1-dimensional hidden layer. After the continuous word embedding input has been transformed, they are fed into a subsequent layer to model the high non-linear relations among words:

$$\mathbf{z} = f(\tilde{E}_{\mathbf{x}}) = f(\mathbf{W}\tilde{E}_{\mathbf{x}} + \mathbf{b}) \tag{6}$$

where $\mathbf{z} \in \mathbb{R}^{d_1}$ is the output of the hidden layer and $\mathbf{b} \in \mathbb{R}^{d_1}$ is a bias vector. $f(\cdot)$ is non-linear function, such as sigmoid function, hyperbolic function, "hard" hyperbolic function, rectifier function, etc. In this paper, we use the rectifier function as our non-linear activation function due to its efficiency and better performance [43]:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

3.1.3. Decoding Process

The decoding process consists of a hidden layer and a reconstruction layer with similar network structures, but different parameters. It transforms the d_1-dimensional vector $f(\tilde{E}_{\mathbf{x}})$ to a md-dimensional vector $g(f(\tilde{E}_{\mathbf{x}}))$. To minimize the reconstruction error with respect to $\tilde{E}_{\mathbf{x}}$, we define the loss function as the L_2 norm of the differences between the original input and the reconstructed input:

$$\mathcal{L}(g(f(\tilde{E}_{\mathbf{x}})), E_{\mathbf{x}}) = ||g(f(\tilde{E}_{\mathbf{x}})) - E_{\mathbf{x}}||_2 \tag{8}$$

We train the neural networks with standard back-propagation algorithm. The gradient of the loss function is computed and back-propagated to the previous layer to update the parameters. Since there are many factors (e.g., learning rate, size of the hidden units) involving in the training process, we will discuss the optimization of these parameters in the later section.

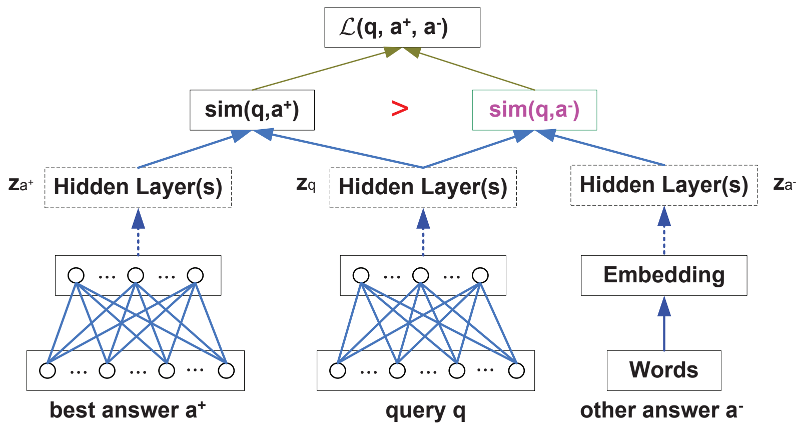

Figure 4. An illustration of the neural network fine-tuning process based on (query, best answer) pairs. In this stage, we use the learned parameters **W** and **b** as initial values, and then minimize the loss function to update the parameters.

3.2. *Fine-Tuning with Question-Answer Pairs*

In the fine-tuning phase, we use another layer on top of the two low-dimensional vectors to maximize the similarity between the queries and the corresponding best answers shown in Figure 4. Since $(query, best\ answer)$ pairs express the similar semantics, the output of the latent representations of a query $f(\mathbf{q})$ and the best answer $f(\mathbf{a})$ should also be close to each other. In fact, the objective of fine-tuning is to discover a latent space which is shared by the queries and the corresponding best answers as much as possible. The shared latent space is particulary useful when the community question answering task tries to match the queries and the answers in the archives.

Given a query-answer pair $< q, a >$, the DAE learns semantic representations for q and a respectively, as $\mathbf{z}_q = f(\mathbf{q})$ and $\mathbf{z}_a = f(\mathbf{a})$ of hidden layer outputs in Figure 4. We then take the two vectors as the input to compute their similarity. Finally, the whole neural network can be fine-tuned towards the supervised criteria with the help of query-answer pairs (a query and the corresponding best answer). The similarity score of the representation pair $< \mathbf{z}_q, \mathbf{z}_a >$ is defined as the cosine similarity of the two vectors:

$$sim(q, a) = \cos(\mathbf{z}_q, \mathbf{z}_a) = \frac{\mathbf{z}_q \cdot \mathbf{z}_a}{||\mathbf{z}_q||||\mathbf{z}_a||} \tag{9}$$

Since $(query, best\ answer)$ pairs express the similar semantics, our goal is to maximize the similarity score between the query part and the best answer part. Inspired by the contrastive estimation method described in [47], we thus propose a supervised training method to learn our model parameters. Let $\mathcal{A}$ denote the set of answers received for query q, for each $(query, best\ answer)$ pair, denoted by (q, a^+) where q is a query and a^+ is the corresponding best answer, we approximate $\mathcal{A}$ by including a^+ and other answers, denoted by $\{a_j^-\}$. We treat $(query, best\ answer)$ pair (q, a^+) as a positive instance and $(query, other\ answer)$ pair (q, a_j^-) as a negative instance. To make the similarity of the positive instance larger than the negative instance, we utilize the following pairwise ranking loss:

$$\mathcal{L}(q, a^+) = \sum_j \max\{0, sim(q, a^+) - sim(q, a_j^-)\} \tag{10}$$

Finally, we minimize the pairwise ranking loss on the query-answer pairs across all training instances:

$$\mathcal{L} = \sum_{<q,a^+>} \mathcal{L}(q, a^+) \tag{11}$$

We use standard back-propagation algorithm to further fine-tune the neural network parameters **W** and **b** (that is, if the learned parameters **W** and **b** in the pre-training phase do not meet the prior knowledge constraint $sim(q, a^+) > sim(q, a^-)$, then the fine-tuning phase will trigger the adjustment of the neural network parameters **W** and **b**.)

Feature	Formulation
F1	$\sum_{q_i \in q \cap a} c(q_i, a)$
F2	$\sum_{q_i \in q \cap a} \log(c(q_i, a) + 1)$
F3	$\sum_{q_i \in q \cap a} \frac{c(q_i,a)}{\lvert a \rvert}$
F4	$\sum_{q_i \in q \cap a} (\frac{c(q_i,a)}{\lvert a \rvert} + 1)$
F5	$\sum_{q_i \in q \cap a} \log(\frac{\lvert C \rvert}{df(q_i)})$
F6	$\sum_{q_i \in q \cap a} (\log(\frac{\lvert C \rvert}{df(q_i)}))$
F7	$\sum_{q_i \in q \cap a} \log(\frac{\lvert C \rvert}{df(q_i)} + 1)$
F8	$\sum_{q_i \in q \cap a} \log(\frac{c(q_i,a)}{\lvert a \rvert})\log(\frac{\lvert C \rvert}{df(q_i)} + 1)$
F9	$\sum_{q_i \in q \cap a} c(q_i, a)\log(\frac{\lvert C \rvert}{df(q_i)})$
F10	$\sum_{q_i \in q \cap a} \log(\frac{c(q_i,a)}{\lvert a \rvert} \frac{\lvert C \rvert}{c(q_i,C)+1} + 1)$
L1	BM25 score

Table 1. LETOR features: $c(t, X)$ is the term frequency of t in X; $df(t)$ is the document frequency of t; $|X|$ is the total number of terms in X; C is the data collection.

4. Scoring Model

Our novel scoring model for community question answering is based on semantic representation of the document texts (e.g., queries and answers), taking into account the statistical-based measures of the similarity between the answer and the query from the retrieved data collection. The relative weight of each component in the scoring formula is determined using a learning to rank (LTR) method based on the query-answer data set. LTR aims at automatically creating the ranking model using training data and machine learning techniques. A typical setting is learning to rank is that feature vectors and ranks are given as training data. A ranking model is learned based on the training data and then applied to the unseen test data. Because of the advantages of it offers, LTR has been gained increasing attention in IR. In this paper, we train the LTR model by using the training data of question-answer pairs. The input to our scoring formula is a query-answer pair (q, a). The first type of features we include is the common statistical-based features, and the second type of feature is our semantic level feature.

4.1. Statistical-Based Features

We use the statistical-based features in the learning to rank (LTR) described in [12]. Table 1 summarizes these features, where L1 is the well known BM25 score [11] of the document as calculated by the search engine for a given query. We use it as one of the baseline features, as well as the one of the baseline scorers to compare with in the experiments.

4.2. Semantic Level Feature

Once the learned parameters are obtained, for each query q (or answer a) from the retrieval data collection, we can transform it into the latent semantic representation

$$\mathbf{z}_q = f(\mathbf{W}\tilde{\mathbf{q}} + \mathbf{b}) \tag{12}$$

where $\mathbf{q}$ is the continuous word embedding vector representation of query q, and $\tilde{\mathbf{q}}$ is the corrupted version of the initial input vector $\mathbf{q}$.

The relevance score between query q and answer a in the semantic space is, then, calculated as the cosine similarity between $\mathbf{z}_q$ and $\mathbf{z}_a$:

$$s(q, a) = \frac{< \mathbf{z}_q, \mathbf{z}_a >}{||\mathbf{z}_q||_2 \cdot ||\mathbf{z}_a||_2} \tag{13}$$

Finally, we experimented with state-of-the-art LTR algorithm SVMRank [13] to determine the final scoring formula. We choose SVMRank [13] as our LTR framework due to its robustness to the training data. For more details, the readers can refer to the related work [13].

5. Experiments

5.1. Data Set

We collect the data set from Yahoo! Answers and use the *getByCategory* function provided in Yahoo! Answers API[4]. The data set used for this paper contains 10 million question pages of Yahoo! Answers (data collection) and 100,000 user queries, each one landed on one of these pages. Our training data set consists of a random selection of 50,000 queries out of the queries in the above data set. On average, each query is associated with one best answer as well as 6.1 other answers, we treat (query, best answer) pair as the positive instance, and all (query, other answer) pairs as the negative instances. Since the cQA collection on which the retrieval task is performed, we use the 10 million Yahoo! Answers data set, indexed by `Lucene`. We use this training data set for learning the parameters used in neural network and the LRT model. Besides, we also sample a held-out validation data of 10,000 query-answer pairs, on which the hyper-parameters are optimized. Note that the training data and the validate data do not intersect with each other.

5.2. Automatic Evaluation

We conduct a large scale automatic evaluation by randomly selecting 10,000 query-answer pairs as a test set, ensuring that each best answer is found among the top 100 retrieved results for its landing query by the BM25 baseline. We also ensure that each query in the test set does not intersect with the queries in the training set mentioned above. For each query, we retrieve the top 100 results from the data collection using `Lucene` with the BM25 scoring function[5]. We use this ranked list of results as our baseline. We then re-rank the list using each of the tested combinations of the statistical-based and semantic level features. This metric evaluates the retrieval performance based on the rank of the best answer in the top 100 ranked list.

5.3. Manual Evaluation

For manual evaluation, we randomly have selected 1,000 queries from the 100,000 user queries. The queries in the test set do not intersect with the queries in the training set. For each query, we collect a pool of 10 answers, constructed from the results retrieved for the query by a variety of ranking methods from our collection. Then, we let three students to judge the relevance of each result in the pool on five Liker scale levels, from non-relevant (-2) to highly relevant (+2). We note that the ranking methods we used for collecting the pool of results do not consider the semantic level features proposed in this work. If the average relevant score of the three students is larger than 0, we will label it as “relevant”; otherwise, we will label it as “non-relevant”. Next, we use the manually judged data set to evaluate the proposed re-ranking scheme. Similar to the automatic evaluation, for each query we retrieved the top 100 results from the collection using `Lucene` with the BM25 scoring function. Then, we re-ranked the list of results using the different tested models.

5.4. Model Architecture

To speed up the training process of the neural network, we implement a distributed framework with mini-batch gradient descent. The gradient descent is learned with the adaptive learning rate procedure called AdaGrad [48]. We use 16 model replicas in each iteration during the training. The parameters are averaged after each iteration and then sent to each replica for the next iteration. The vocabulary size V for the input vector is 20,000 and we choose different lengths for the hidden layer as $d_1 = \{100, 200, 300, 400, 500, 600\}$ in the experiments. In the pre-training phase, all query-answer pairs in the training data set is fed into two neural networks respectively for DAE training, where network parameters $\mathbf{W}$ and $\mathbf{b}$ are randomly initialized. We choose the constant learning rate with different values $\{0.1, 0.01, 0.001, 0.0001\}$ and finally select the model with the lowest reconstruction error. In the fine-tuning phase, we also choose a constant learning rate in the same range as the pre-training rate with respect to the supervised fine-tuning validate error.

[4]http://developer.yahoo.com/answers

[5]We use the BM25 implementation provided by Apache Lucene (http://lucene.apache.org/), using the default parameter setting ($k_1 = 1.2$, $b = 0.75$).

#	Model	MRR	R@1	R@5	R@10
1	BM25	0.372	0.285	0.446	0.576
2	BM25+*letor*	0.395	0.297	0.448	0.602
3	BM25+sf	0.418	0.310	0.502	0.627
4	BM25+*letor* + sf	**0.425**	**0.321**	**0.526**	**0.643**
5	TR	0.381	0.289	0.460	0.583
6	TRLM	0.403	0.302	0.492	0.615
7	PTRLM	0.409	0.306	0.495	0.620
8	UQATM	0.394	0.296	0.483	0.595
9	SQATM	0.405	0.305	0.493	0.617

Table 2. Experimental results for the automatic evaluation. The bold formate indicates that the difference between the results of our proposed approach (*all* model) and other methods are statistically significant with $p < 0.05$ under a t-test.

5.5. Ranking Models and Comparison Methods

In order to validate the effectiveness of the proposed approach, we evaluate the learning to rank models using the two combination of features.

- BM25: This is the baseline model with BM25 scoring function provided by `Lucene`.
- BM25+*letor*: Adds to the other 10 LETOR features [12] to the baseline BM25 score (see Table 1).
- BM25+sf: Adds to the semantic level feature to the baseline BM25 score.
- BM25+*letor* + sf: Adds to the baseline BM25 score all the statistical-based and semantic level features.

For each combination of features, we train a separate ranking model using SVMRank, based on the training data set described in section 5.1. We evaluate the different models under two test sets: the first is based on a large-scale query-answer pairs and the second is based on manual judgements. Since our proposed semantic level feature aims at solving the lexical gap between the queries and the answers, we also compare our approach with the two groups of methods which also address the lexical gap in the literature. The first group is the translation models: word-based translation model (TR) [1], word-based translation language model (TRLM) [2] and phrase-based translation language model (PTRLM) [8]. We implement those three translation models based on the original papers and train those model with (query, best answer) pairs from the training data set mentioned above. The second group is the topic-based methods: unsupervised query-answer topic model (UQATM) [26] and supervised question-answer pair topic model (SQATM) [9]. UQATM can learn the latent topics aligned across the question-answer pairs to alleviate the lexical gap problem, with the assumption that a question and its paired answer share the same topics distribution. SQATM assumes that questions and answers share some common latent topics and are generated in a “question part” and “answer part” respectively following the topics. The topics also determine an answer quality signal. The advantage of this approach not only comprehensively models user behaviors on cQA sites, but also highlights the instinctive heterogeneity of questions and answers. We re-implement these two topic-based models and tune the parameter settings on our data set.

5.6. Experimental Results

In this subsection, we present the results on the automatic evaluation and manual evaluation data sets. For the automatic evaluation test set, we test the performance of retrieved results of the various ranking models using MRR, Binary-Recall $R@N$ (the relative number of queries with $P@N > 0$) and MAP [49]:

- **MAP (Mean Average Precision):** For a set of queried questions Q, MAP measures the mean of the average precision for each queried question $\mathbf{q}$ for a method $\mathcal{M}$:

$$\text{MAP} = \frac{\sum_{\mathbf{q}\in Q} \text{Avg}P(\mathbf{q})}{|Q|} \tag{14}$$

#	Model	MAP	P@1	P@3	P@5	P@10	R@1	R@3	R@5	R@10
1	BM25	0.247	0.213	0.165	0.112	0.085	0.213	0.326	0.363	0.405
2	BM25+*letor*	0.260	0.226	0.177	0.123	0.093	0.226	0.345	0.371	0.432
3	BM25+*sf*	0.268	0.231	0.180	0.128	0.096	0.231	0.362	0.382	0.446
4	BM25+*letor* + *sf*	**0.272**	**0.235**	**0.183**	**0.135**	**0.096**	**0.235**	**0.370**	**0.388**	**0.453**
5	TR	0.256	0.225	0.175	0.117	0.087	0.225	0.338	0.365	0.427
6	TRLM	0.262	0.229	0.178	0.125	0.094	0.229	0.351	0.376	0.438
7	PTRLM	0.265	0.230	0.178	0.126	0.094	0.230	0.355	0.377	0.441
8	UQATM	0.258	0.226	0.177	0.119	0.089	0.227	0.344	0.308	0.431
9	SQATM	0.263	0.231	0.177	0.124	0.093	0.230	0.353	0.376	0.439

Table 3. Experimental results for the manual evaluation. The bold font indicates that the difference between the results of our proposed approach (*all* model) and other methods are statistically significant with $p < 0.05$ under a t-test.

$$\mathrm{Avg}P(\mathbf{q}) = \frac{1}{N_{\mathcal{M}_\mathbf{q}}} \sum_{j=1}^{|\mathcal{M}_\mathbf{q}|} \frac{N_{\mathcal{M}_{\mathbf{q},j}}}{j} \mathbf{1}(\mathcal{M}_{\mathbf{q},j}) \tag{15}$$

where $\mathbf{1}(\mathcal{S})$ is an indicator function which returns 1 when $\mathcal{M}_{\mathbf{q},j}$ is relevant based on the test collection we constructed. $N_{\mathcal{M}_{\mathbf{q},j}}$ denotes the number of relevant questions among the top j ranked list returned by $\mathcal{M}$ for queried question $\mathbf{q}$, and $N_{\mathcal{M}_\mathbf{q}}$ denotes the total number of relevant questions of queried question $\mathbf{q}$ returned by a method $\mathcal{M}$, and $\mathcal{M}_{\mathbf{q},j}$ is the j-th question generated by method $\mathcal{M}$ for queried question $\mathbf{q}$. MAP rewards methods that return relevant questions early and also rewards correct ranking of the results.

- *P@N* **(Precision** *@N***):** For a set of queried questions Q, $P@N$ is the fraction of the top N retrieved questions that are relevant to the queried questions for a method $\mathcal{M}$:

$$P@N = \frac{1}{|Q|} \sum_{\mathbf{q} \in Q} \frac{N_{\mathcal{M}_{\mathbf{q},N}}}{N} \tag{16}$$

where $N_{\mathcal{M}_{\mathbf{q},N}}$ denotes the number of relevant questions among the top N ranked list returned by a method $\mathcal{M}$ for queried question $\mathbf{q}$.

- *MRR* **(Mean Reciprocal Rank):** MRR is a statistical measure for evaluating any process that produces a list of possible responses to a set of queries, ordered by probability of correctness. The reciprocal rank of a query response is the multiplicative inverse of the rank of first correct answer. The mean reciprocal rank is the average of the reciprocal ranks of results for a set of queries Q:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\mathrm{rank}_i} \tag{17}$$

The reciprocal value of the mean reciprocal rank corresponds to the harmonic mean of the ranks.

Table 2 presents the results of the automatic evaluation. From this table, we can see that *letor* features within a learning to rank framework improve over the basic `Lucene` BM25 scoring function. In our test set, the improvement of incorporating the *letor* statistical-based features is 2.3% for MRR, and for $R@K$ ranges from 1.2% ($R@1$) and up to 2.6% ($R@10$).

Then, we input the semantic level features into learning to rank framework without considering the statistical-based features. We find that it obtains a much larger improvement over the baseline BM25 score, for example, MRR is increased by 4.6%, $R@10$ is increased by 5.1%. In this experiment, it seems that semantic level feature provides more useful information for ranking than statistical-based features, thereby achieving higher results in all ranking measures.

Finally, looking at our *all* model, we can see that semantic level feature provides some complementary information to that of statistical-based features. This model consistently improves over all other models, including the well known translation models [1, 2, 8]. For example, the improvement in MRR compared to the baseline BM25 is 5.3%, a relative

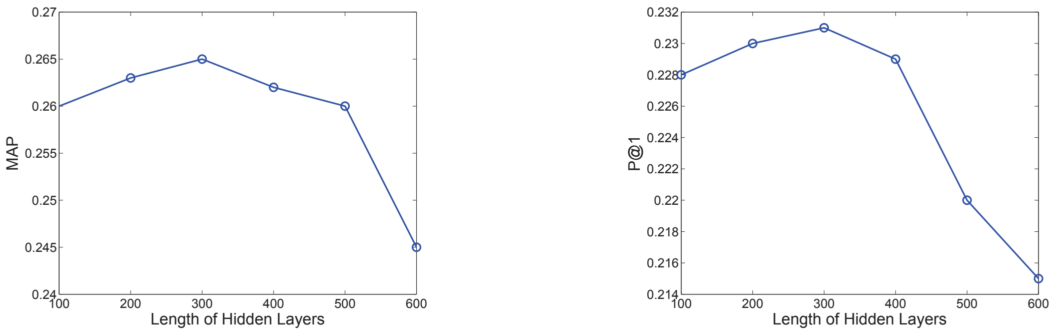

Figure 5. The influence on different length of hidden layers for community question answering on the manual evaluation.

increase of 3.91% compared to the improvement achieved using phrase-based translation model [8]. We also conduct a significant test, all differences are statistically significant with $p < 0.05$.

Under the manual evaluation, where gold standard annotations are provided, we evaluate the performance of each tested ranking model using Mean Average Precision (MAP), Precision and Binary-Recall at K ($P@K$, $R@K$) [49]. Table 3 shows the experimental results on the manual evaluation. From this table, we can see the usefulness of the semantic level feature in this test set. For example, MAP is improved by 2.1%, $P@10$ is increased by 1.1% and $R@10$ is increased by 4.1% over the baseline BM25 scoring function. Combining all features together (*all* model) can still gain an improvement over the baseline BM25, but it has no conclusive improvement over just taking a semantic level feature. We also note that the differences between *all* and translation models [1, 2, 8] in all measures are statistically significant at $p < 0.05$.

The reason for this difference between the automatic evaluation and the manual evaluation may be that in automatic evaluation, query and the corresponding best answer are sought. These are answers that are most likely shown to web searchers as part of the top results for the issued queries. If we assume that search engines utilize mainly statistical-based features in their ranking function, then the retrieval results, from which users choose the best answers, introduce a bias towards statistical-based features. It encourages to incorporate semantic level features that contribute to a significant improvement in performance even with this bias. In our manual evaluation, all top results retrieved by the different models are evaluated, therefore reducing this bias.

5.7. *Effect of Length of Hidden Layers*

The length of hidden layers d_1 in the neural network is an important factor for community question answering retrieval. In deep learning, this parameter is often empirically tuned with human efforts. In this paper, we look into how the hidden layers influence the final retrieval performance. As shown in Figure 5, the retrieval performance is better when d_1 is relatively small. Actually, there is no obvious difference of the performance when d_1 is less than 400. However, when d_1 equals 600, the retrieval performance is inferior to other settings. The main reason is that parameters in the neural networks are too many to be effectively trained. As we know when $d_1 = 600$, there are a total of 600×600 parameters between the linear and non-linear layers in the neural network. Limited training data prevents the model from getting close to the global optimum. Therefore, the model is likely to fall in the local optima and lead to unacceptable representations.

6. Conclusion

This paper presents a novel approach to learn the semantic representation for queries and answers by using a neural network architecture. The network establishes the semantic relationship for query-answer pairs by minimizing the reconstruction and pairwise ranking errors. Then, the semantic level features are incorporated into a LTR framework, as well as a set of statistical-based features. The importance of each feature compared to the rest is learned via a SVMRank algorithm. Experiments conducted on both automatic evaluation and manual evaluation show a significant improvement in community question answering retrieval. An open question which our work does not deal with is how answer quality affects the effectiveness of our ranking approach, we leave it for future research.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61303180 and No. 61573163), the Beijing Natural Science Foundation (No. 4144087), and also Sponsored by CCF-Tencent Open Research Fund. We thank the anonymous reviewers for their insightful comments.

References

[1] J. Jeon, W. B. Croft, J. H. Lee, Finding Similar Questions in Large Question and Answer Archives, in: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, 2005, pp. 84–90.

[2] X. Xue, J. Jeon, W. B. Croft, Retrieval Models for Question and Answer Archives, in: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2008, pp. 475–482.

[3] G. Zhou, Y. Chen, D. Zeng, J. Zhao, Group non-negative matrix factorization with natural categories for question retrieval in community question answer archives, in: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, Dublin City University and Association for Computational Linguistics, Dublin, Ireland, 2014, pp. 89–98.

[4] H. Duan, Y. Cao, C. yew Lin, Y. Yu, Searching Questions by Identifying Question Topic and Question Focus, in: Proceedings of 46th Annual Meeting of the Association for Computational Linguistics, 2008.

[5] J.-T. Lee, S.-B. Kim, Y.-I. Song, H.-C. Rim, Bridging Lexical Gaps Between Queries and Questions on Large Online Q&A Collections with Compact Translation Models, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2008, pp. 410–418.

[6] D. Bernhard, I. Gurevych, Combining Lexical Semantic Resources with Question & Answer Archives for Translation-based Answer Finding, in: Proceedings of the ACL-IJCNLP, 2009, pp. 728–736.

[7] X. Cao, G. Cong, B. Cui, C. S. Jensen, A Generalized Framework of Exploring Category Information for Question Retrieval in Community Question Answer Archives, in: Proceedings of the 19th International Conference on World Wide Web, 2010.

[8] G. Zhou, L. Cai, J. Zhao, K. Liu, Phrase-based Translation Model for Question Retrieval in Community Question Answer Archives, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, 2011, pp. 653–662.

[9] K. Zhang, W. Wu, H. Wu, Z. Li, M. Zhou, Question retrieval with high quality answers in community question answering, in: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014, 2014, pp. 371–380.

[10] D. Carmel, A. Mejer, Y. Pinter, I. Szpektor, Improving Term Weighting for Community Question Answering Search Using Syntactic Analysis, in: Proceedings of the 23rd ACM International Conference on Information and Knowledge Management, 2014, pp. 351–360.

[11] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, M. Gatford, Okapi at Trec-3, in: TREC, 1994, pp. 109–126.

[12] T. Qin, T.-Y. Liu, J. Xu, H. Li, LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval, Inf. Retr. 13 (4).

[13] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, H.-W. Hon, Adapting Ranking SVM to Document Retrieval, in: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2006.

[14] W.-t. Yih, K. Toutanova, J. C. Platt, C. Meek, Learning Discriminative Projections for Text Similarity Measures, in: Proceedings of the 15th Conference on Computational Natural Language Learning, 2011, pp. 247–256.

[15] A. Singh, Entity Based q&a Retrieval, in: EMNLP-CoNLL, 2012, pp. 1266–1277.

[16] G. Zhou, S. Lai, K. Liu, J. Zhao, Topic-sensitive probabilistic model for expert finding in question answer communities, in: 21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012, 2012, pp. 1662–1666.

[17] G. Zhou, Y. Chen, D. Zeng, J. Zhao, Towards faster and better retrieval models for question search, in: Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management, CIKM '13, 2013, pp. 2139–2148.

[18] G. Zhou, J. Zhao, T. He, W. Wu, An empirical study of topic-sensitive probabilistic model for expert finding in question answer communities, Knowledge Based Systems (2014) 136–145.

[19] W. Zhang, Z. Ming, Y. Zhang, T. Liu, T. Chua, Exploring key concept paraphrasing based on pivot language translation for question retrieval, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA., 2015, pp. 410–416.

[20] G. Zhou, T. He, J. Zhao, P. Hu, Learning continuous word embedding with metadata for question retrieval in community question answering, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, Beijing, China, 2015, pp. 250–259.

[21] X. Cao, G. Cong, B. Cui, C. S. Jensen, C. Zhang, The use of categorization information in language models for question retrieval, in: Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009, 2009, pp. 265–274.

[22] L. Cai, G. Zhou, K. Liu, J. Zhao, Learning the Latent Topics for Question Retrieval in Community QA, in: Proceedings of 5th International Joint Conference on Natural Language Processing, 2011, pp. 273–281.

[23] S. Riezler, E. Vasserman, I. Tsochantaridis, V. Mittal, Y. Liu, Statistical Machine Translation for Query Expansion in Answer Retrieval, in: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, 2007.

[24] G. Zhou, K. Liu, J. Zhao, Exploiting bilingual translation for question retrieval in community-based question answering, in: Proceedings of COLING 2012, The COLING 2012 Organizing Committee, Mumbai, India, 2012, pp. 3153–3170.

[25] G. Zhou, F. Liu, Y. Liu, S. He, J. Zhao, Statistical machine translation improves question retrieval in community question answering via matrix factorization, in: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Sofia, Bulgaria, 2013, pp. 852–861.

[26] Z. Ji, F. Xu, B. Wang, B. He, Question-answer Topic Model for Question Retrieval in Community Question Answering, in: Proceedings of CIKM, 2012, pp. 2471–2474.

[27] J. Guo, S. Xu, S. Bao, Y. Yu, Tapping on the potential of q&a community by recommending answer providers, in: Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008, 2008, pp. 921–930.

[28] K. Wang, Z. Ming, T.-S. Chua, A Syntactic Tree Matching Approach to Finding Similar Questions in Community-based Qa Services, in: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2009, pp. 187–194.

[29] B. Wang, B. Liu, X. Wang, C. Sun, D. Zhang, Deep learning approaches to semantic relevance modeling for chinese question-answer pairs, ACM Trans. Asian Lang. Inf. Process. 10 (4) (2011) 21.

[30] H. Hu, B. Liu, B. Wang, M. Liu, X. Wang, Multimodal dbn for predicting high-quality answers in cqa portals, in: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Sofia, Bulgaria, 2013, pp. 843–847.

[31] Z. Lu, H. Li, A deep architecture for matching short texts, in: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States., 2013, pp. 1367–1375.

[32] M. Iyyer, J. Boyd-Graber, L. Claudino, R. Socher, H. Daumé III, A neural network for factoid question answering over paragraphs, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 2014, pp. 633–644.

[33] B. Hu, Z. Lu, H. Li, Q. Chen, Convolutional neural network architectures for matching natural language sentences, in: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, 2014, pp. 2042–2050.

[34] X. Qiu, X. Huang, Convolutional neural tensor network architecture for community-based question answering, in: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, 2015, pp. 1305–1311.

[35] M. Surdeanu, M. Ciaramita, H. Zaragoza, Learning to Rank Answers on Large Online QA Collections, in: ACL, 2008, pp. 719–727.

[36] B. Wang, X. Wang, C. Sun, B. Liu, L. Sun, Modeling Semantic Relevance for Question-Answer Pairs in Web Social Communities, in: ACL, 2010, pp. 1230–1238.

[37] D.-R. Liu, Y.-H. Chen, C.-K. Huang, Qa document recommendations for communities of question-answering websites, Know.-Based Syst. 57 (2014) 146–160.

[38] B. Song, Z. Jiang, X. Li, Modeling knowledge need awareness using the problematic situations elicited from questions and answers, Know.-Based Syst. 75 (C) (2015) 173–183.

[39] M. Romero, A. Moreo, J. L. Castro, A cloud of faq: A highly-precise faq retrieval system for the web 2.0, Know.-Based Syst. 49 (2013) 81–96.

[40] A. Moreo, E. M. Eisman, J. L. Castro, J. M. Zurita, Learning regular expressions to template-based faq retrieval systems, Know.-Based Syst. 53 (2013) 108–128.

[41] G. Zhou, Y. Liu, F. Liu, D. Zeng, J. Zhao, Improving question retrieval in community question answering using world knowledge, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13, AAAI Press, 2013, pp. 2239–2245.

[42] Y. Bengio, Learning deep architectures for ai, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.

[43] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, U. D. Montral, M. Qubec, Greedy Layer-Wise Training of Deep Networks, in: NIPS, 2007.

[44] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and Composing Robust Features with Denoising Autoencoders, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 1096–1103.

[45] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, L. Heck, Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data, in: Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, 2013, pp. 2333–2338.

[46] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of NIPS, 2013, pp. 3111–3119.

[47] N. A. Smith, J. Eisner, Contrastive Estimation: Training Log-linear Models on Unlabeled Data, in: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, 2005.

[48] J. Duchi, E. Hazan, Y. Singer, Adaptive Subgradient Methods for Online Learning and Stochasitc Optimization, J. Mach. Learn. Res. 12 (2011) 2121–2159.

[49] R. A. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2011.