Learning Distributed Representations of Data in Community Question Answering for Question Retrieval

Kai Zhang [*]
State Key Laboratory of Software Development Environment
Beihang University, China
zhangkai@buaa.edu.cn

Wei Wu
Microsoft Research
Beijing, China
wuwei@microsoft.com

Fang Wang
State Key Laboratory of Software Development Environment
Beihang University, China
fangwang@buaa.edu.cn

Ming Zhou
Microsoft Research
Beijing, China
mingzhou@microsoft.com

Zhoujun Li
State Key Laboratory of Software Development Environment
Beihang University, China
lizj@buaa.edu.cn

ABSTRACT

We study the problem of question retrieval in community question answering (CQA). The biggest challenge within this task is lexical gaps between questions since similar questions are usually expressed with different but semantically related words. To bridge the gaps, state-of-the-art methods incorporate extra information such as word-to-word translation and categories of questions into the traditional language models. We find that the existing language model based methods can be interpreted using a new framework, that is they represent words and question categories in a vector space and calculate question-question similarities with a linear combination of dot products of the vectors. The problem is that these methods are either heuristic on data representation or difficult to scale up. We propose a principled and efficient approach to learning representations of data in CQA. In our method, we simultaneously learn vectors of words and vectors of question categories by optimizing an objective function naturally derived from the framework. In question retrieval, we incorporate learnt representations into traditional language models in an effective and efficient way. We conduct experiments on large scale data from Yahoo! Answers and Baidu Knows, and compared our method with state-of-the-art methods on two public data sets. Experimental results show that our method can significantly improve on baseline methods for retrieval relevance. On 1 million training data, our method takes less than 50 minutes to learn a model on a single multicore machine, while the translation based language model needs more than 2 days to learn a translation table on the same machine.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval models*

Keywords

community question answering; question retrieval; word vector representation; unsupervised model

1. INTRODUCTION

Community question answering (CQA) portals, like Yahoo! Answers and Baidu Knows, have became popular platforms for people to gain knowledge from others. The knowledge is organized as questions posted by askers and lists of answers provided by answerers. There is also a significant amount of metadata associated with questions and answers. For example, askers choose categories for their questions, and award best answer marks to answerers whose answers are the most informative or satisfying. The data has made CQA portals valuable resources for many tasks like information retrieval, data mining, and question answering, etc.

One fundamental task for leveraging the knowledge in CQA is finding similar questions for new queries (a.k.a. question retrieval). Many studies have been done along this line [10, 26, 5, 24, 4, 29]. The biggest challenge in question retrieval is that users often express similar meanings with different but semantically related words, which generates lexical gaps between similar questions. For example, for the query "what are good foods for a gymnast to eat", there is a similar question "Im a gymnast, what should my meal plan be".The two questions have similar meanings but share very few common words, which prevents traditional retrieval models like BM25 [19], and language models for information retrieval (LMIR) [18, 27] from recognizing their similarity. To bridge the lexical gaps, state-of-the-art methods leverage

[*]The work was done when the first author was an intern at Microsoft Research Asia.

WSDM 2016, February 22-25, 2016, San Francisco, CA, USA
 ISBN 978-1-4503-3716-8/16/02 ...$15.00.
DOI: http://dx.doi.org/10.1145/2835776.2835786.

word-to-word (or phrase-to-phrase) translation probabilities [10, 26] or question categories [5, 4], and expand the traditional language models with semantic matching items.

Existing models, although proposed from different background, can be interpreted by a unified framework. In fact, we find that traditional language models (LMIR) [18, 27] , translation based language models (TRLM) [26], and language models with category smoothing (LMC) [5] match questions in a similar manner, that is they represent words and question categories as elements in a vector space and linearly combine the dot products of the vectors as a similarity function. The advantage of taking this view is that it provides us with a new perspective to understand existing models and inspires us to create new models. From the framework, we find that the key to measuring question similarity is how to represent the data in CQA. Existing models, however, are either heuristic on data representation or difficult to scale up. For example, in LMIR and LMC, words are represented as indices in a vocabulary and similarity of words is ignored entirely. Although in LMC a category vector is considered for matching semantically similar questions, it is still not clear how semantic relations of words are modeled. Translation based language models go beyond the simple word-level matching, and model the semantic relations of words by mapping the word vectors in LMIR to a new vector space with a word-word translation matrix. The models, however, rely heavily on translation probabilities whose estimation needs a cubic time algorithm with respect to the vocabulary size [10], and thus are difficult to scale up. Since existing models suffer from such problems, a natural question arises: whether we can automatically learn representations of words and categories in CQA for question retrieval. The learning approach must appropriately model the relations of words, and be unsupervised and efficient enough to leverage the big data in CQA.

In this paper, we propose learning representations of data in CQA and leveraging the representations for question retrieval. The learning approach is conducted by optimizing an objective naturally derived from the framework. The objective applies a non-linear transformation to the linear combination of dot products of word vectors and question category vectors. In learning, we consider local relatedness of words. A word is only related to a question category and words from a small window in a question or in the associated answers. Our method simultaneously embeds words in questions, words in answers, and question categories into a vector space, which extends the recently proposed continuous bag-of-words (CBOW) architecture [12, 15, 7] for learning distributed representations for words in machine learning. We benefit from the architecture by a negative sampling technique and an optimization algorithm that runs in linear time with respect to number of words in training data and is independent of vocabulary size. The implementation code is published at `https://github.com/ComputerHobbyist/cqa`.

We create a retrieval model by incorporating the representations of data into traditional language models. Specifically, we take category vectors as centroids of clusters, and partition the vocabulary by assigning each word to the cluster of its most similar category. In each cluster, we calculate the similarity of each word and its top N similar words. The similarities are normalized as probabilities. We replace the word-word translation probability in TRLM with the probability calculated using word vectors. The "new" TRLM is further combined with the similarity of a word vector and a category vector as the final retrieval model. The model enjoys many advantages. First, since a word is only related to words from the same category, the category vector can help filter noise in word-word similarity. Second, when calculating word-word similarity, we only care about words in a small cluster. This can greatly enhance model estimation efficiency. Third, since we only keep nearest neighbors for each word, online memory cost only linearly increases with vocabulary size. The new retrieval model can be integrated into a learning to rank system as a feature to improve the performance of the system on question ranking.

We conduct experiments on large scale data from Yahoo! Answers and Baidu Knows, and compare our method with state-of-the-art methods on data sets published at `https://github.com/ComputerHobbyist/cqa`. On 1 million training data, the translation based language model needs nearly 2 days to learn a translation table on a single multicore machine, while on the same machine, our method takes less than 50 minutes to learn a model and significantly improves upon baseline methods of retrieval relevance.

Our contributions in the paper are three-fold: (1) proposal of a unified framework which provides a new perspective for existing retrieval models; (2) proposal of a new embedding based retrieval model; (3) empirical verification of the proposed model on both efficacy and efficiency.

2. RELATED WORK

Together with the flourishing of community question answering (CQA) archives, research on question retrieval in CQA has received much attention. State-of-the-art methods are based on language models [18, 27]. For example, Cao et al. [5, 4] propose a language model with category smoothing in which they estimate a new smoothing item from questions under the same category. Jeon el al. [10] propose learning word-to-word translation probabilities from question-question pairs collected based on similar answers. Xue et al. [26] learn a translation model from question-answer pairs. In this paper, we interpret language model based methods from an embedding perspective, and find that existing models suffer from efficacy and efficiency problems. Inspired by this perspective, we propose an approach that automatically learns the representations of data in an efficient way. The learnt representations are incorporated into LMIR to create a new model that is proven to be more effective in question retrieval through public data sets.

Learning distributed representations for words with neural network architectures has become a hot research topic in recent years [2, 17, 16, 13, 8, 12, 15, 11, 7], and has proven successful on speech recognition, image annotation, machine translation and other NLP tasks [21, 20, 6, 25, 14]. For statistical language modeling, Bengio et al. [2] proposed a neural network language model (NNLM) which simultaneously learns a distributed representation for each word and the probability function for word sequences expressed by the representations. The model obtains better generalization ability from the distributed representations of words, and is proven to be more effective than n-gram language models. To enhance the efficiency of NNLM, Morin et al. [17] introduced a hierarchical softmax method as an approximation of the full softmax in NNLM. Recently, Mikolov et al. [12, 15] have simplified the architecture of NNLM, and proposed continuous bag-of-words (CBOW) and Skip-gram

models that can further speed up the learning of word representations. They show that the learning approach is surprisingly good at capturing the semantic relations of words. Le and Mikolov further extend the architectures of CBOW and Skig-gram and learn a paragraph vector to represent variable-length pieces of texts [11]. In this work, we learn distributed representations of data in CQA for question retrieval. Our learning method extends the CBOW architecture by simultaneously embedding words in questions, words in answers, and question categories into a vector space.

Before our work, there have already existed some studies on building deep architectures for searching. For example, Huang et al. [9] propose a deep structured semantic model (DSSM) for matching queries and documents. Shen et al. [22] follow the idea of DSSM and proposed a convolutional latent semantic model (CLSM) to express n-gram level and sentence level contextual structures in matching. The problem of applying DSSM and CLSM to question retrieval in CQA is that both of them need large scale supervision signals for training, which are not easy to obtain in CQA. Contrary to DSSM and CLSM, our model is totally unsupervised, and thus can leverage big data from CQA to learn semantic similarities between questions.

3. PRELIMINARIES

Before presenting the new framework, we first give an overview of state-of-the-art retrieval models in CQA.

3.1 Language Models for Information Retrieval

Recently, language models for information retrieval (LMIR) [18, 27] and their extensions have proven effective for question retrieval in CQA. Formally, given a query question q and a candidate question Q, LMIR calculates the similarity between q and Q by

$$P(q|Q) = \prod_{w \in q} [(1-\lambda)P_{ml}(w|Q) + \lambda P_{ml}(w|C)], \quad (1)$$

where $P_{ml}(w|Q)$ represents the maximum likelihood of a word w estimated from Q, and $P_{ml}(w|C)$ is a smoothing item which is calculated as the maximum likelihood in a large corpus C. The smoothing item avoids zero probability which stems from the words appearing in the query but not in the candidate. $\lambda \in (0,1)$ is a parameter which acts as a trade-off between the likelihood and the smoothing item.

3.2 Language Models with Category Smoothing

In most CQA portals, each question is tagged with one or several categories by askers. The categories can be viewed as explicit topics of questions, and thus can reflect semantic similarities between questions. To leverage question categories, Cao et al. [5] propose a language model with category smoothing (LMC) by expanding LMIR with a new smoothing item estimated from questions under the same category. The underlying idea is that questions under the same category are similar at a topic level. The model of LMC is defined as

$$P(q|Q) = \prod_{w \in q} [(1-\lambda)P_{ml}(w|Q) + \lambda P_s(w|Q)], \quad (2)$$

where

$$P_s(w|Q) = (1-\beta)P_{ml}(w|C) + \beta P_{ml}(w|Cat(Q)).$$

Here $\beta \in (0,1)$ is a parameter. $Cat(Q)$ denotes the category of candidate question Q, which is usually a root category or a leaf category in the category hierarchy of a CQA portal. $P_{ml}(w|Cat(Q))$ is the maximal likelihood of word w estimated from $Cat(Q)$. Note that in Equation (2), we do not involve the question classification item as in the work of Cao el al. [5], since the item only sightly improves the model from empirical results [5].

3.3 Translation-based Language Model

Translation based language models (TRLM) [3] leverage word-to-word or phrase-to-phrase translation probabilities estimated from questions and answers to match semantically similar questions. TRLM can be formulated as

$$P(q|Q) = \prod_{w \in q} [(1-\lambda)P_{mx}(w|Q) + \lambda P_{ml}(w|C)], \quad (3)$$

where

$$P_{mx}(w|Q) = (1-\beta)P_{ml}(w|Q) + \beta P_{tr}(w|Q)$$
$$P_{tr}(w|Q) = \sum_{v \in Q} P_{tp}(w|v)P_{ml}(v|Q).$$

Here λ and β are parameters. $P_{tp}(w|v)$ represents the translation probability from word v in candidate question Q to word w in query question q.

Xue, et al. [26] further improved TRLM by introducing an extra answer likelihood item $P_{ml}(w|a)$ estimated from an answer a of question Q. The model can be represented as

$$P(q|Q,a) = \prod_{w \in q} [(1-\lambda)P_{mx}(w|Q,a) + \lambda P_{ml}(w|C)] \quad (4)$$

where

$$P_{mx}(w|Q,a) = \alpha P_{ml}(w|Q) + \beta P_{tr}(w|Q) + \gamma P_{ml}(w|a).$$

Here α, β and γ are parameters satisfying $\alpha + \beta + \gamma = 1$. We denote the model in Equation (4) as TAL.

4. A FRAMEWORK FOR RETRIEVAL MODELS IN CQA

In this section, we interpret the existing language model based methods described in Section 3 with a new framework. We find that these models perform question matching in a similar manner, that is they represent words in questions, words in answers, and question categories as vectors in a vector space and calculate question similarity by dot products of the vectors. The framework exposes the essence of question retrieval to us and inspires us to create better retrieval models.

From Equation (1), (2), (3), and (4), we can see that the existing retrieval models can be represented as

$$P(q|Q) = \prod_{w \in q} P(w|Q),$$

where $P(q|Q)$ is the generation probability of a query q from a candidate Q, and $P(w|Q)$ is the generation probability of a word w from Q. Note that in LMC and TAL, Q not only means the question itself, but also contains metadata associated with the question. In LMC, the metadata is the category of Q, while in TAL, it is the answer of Q. We can understand $P(q|Q)$ and $P(w|Q)$ as similarity functions between q and Q, and w and Q, respectively. When performing

question matching, all the existing models decompose the question-question similarity to a product of several word-question similarity functions. The difference between them is how they define and estimate the word-question similarity function.

For LMIR, if we define $v_{ohs}(w)$ as the one-hot-spot representation of a word w in which only one element is equal to one indicating the index of w in the vocabulary and all other elements are zero, then $P_{ml}(w|Q)$ in Equation (1) can be re-written as

$$P_{ml}(w|Q) = \frac{1}{L_Q} v_{ohs}(w) \cdot \sum_{w' \in Q} v_{ohs}(w'),$$

where L_Q is the length of Q. Since $P_{ml}(w|C)$ in Equation (1) is independent of Q, we denote it as $b(w)$. Thus, the word-question similarity for LMIR can be expressed as

$$\frac{1-\lambda}{L_Q} v_{ohs}(w) \cdot \sum_{w' \in Q} v_{ohs}(w') + \lambda b(w).$$

For LMC, $P_{ml}(w|Cat(Q))$ in Equation (2) can be re-written as

$$P_{ml}(w|Cat(Q)) = \frac{1}{L_{cat}} v_{ohs}(w) \cdot \sum_{w' \in Cat(Q)} v_{ohs}(w'),$$

where L_{cat} is number of words in $Cat(Q)$. If we understand $\sum_{w' \in Cat(Q)} v_{ohs}(w')$ as a vector representation for category $Cat(Q)$, and denote it as $v(Cat(Q))$, then the word-question similarity function for LMC can be represented as

$$\alpha_1 v_{ohs}(w) \cdot \sum_{w' \in Q} v_{ohs}(w') + \alpha_2 v_{ohs}(w) \cdot v(Cat(Q)) + \alpha_3 b(w),$$

where $\alpha_1 = \frac{1-\lambda}{L_Q}$, $\alpha_2 = \frac{\lambda\beta}{L_{cat}}$, and $\alpha_3 = \lambda(1-\beta)$.

For TRLM and TAL, we denote the translation matrix as $M_{tr} = (P_{tp}(w|v))_{n \times n}$ and the length of answer a as L_a, then $P_{tr}(w|Q)$ in Equation (3) and $P_{ml}(w|a)$ in Equation (4) can be re-written as

$$P_{tr}(w|Q) = \frac{1}{L_Q} [M_{tr}^\top v_{ohs}(w)] \cdot \sum_{w' \in Q} v_{ohs}(w')$$

$$P_{ml}(w|a) = \frac{1}{L_a} v_{ohs}(w) \cdot \sum_{w' \in a} v_{ohs}(w').$$

The word-question similarity for TRLM can be represented as

$$[\alpha_1 I + \alpha_2 M_{tr}^\top] v_{ohs}(w) \cdot \sum_{w' \in Q} v_{ohs}(w') + \alpha_3 v_{ohs}(w) \cdot \sum_{w' \in a} v_{ohs}(w') + \alpha_4 b(w),$$

where $\alpha_1 = \frac{(1-\lambda)\alpha}{L_Q}$, $\alpha_2 = \frac{(1-\lambda)\beta}{L_Q}$, $\alpha_3 = \frac{(1-\lambda)\gamma}{L_a}$, and $\alpha_4 = \lambda$.

From the analysis above, we find that the existing retrieval models share some common characteristics in calculating word-question similarity: (1) they represent words in questions, words in answers, and question categories as vectors; (2) they calculate dot products of the vectors and linearly combine the dot products. These models, therefore, can be viewed as special cases of the following general framework for question retrieval:

$$f(q, Q) = \prod_{w \in q} g(w, Q) \tag{5}$$

$$g(w, Q) = \alpha_1 A v(w) \cdot \sum_{w' \in Q} v(w') + \alpha_2 v(w) \cdot v\left(Cat(Q)\right) + \alpha_3 v(w) \cdot \sum_{w' \in a} v(w') + \alpha_4 b(w). \tag{6}$$

Here, we use $f(q, Q)$ and $g(w, Q)$ to represent general similarity functions for a question pair and a word-question pair, respectively. $v(w)$ and $v\left(Cat(Q)\right)$ represent vector representations of a word and a question category, respectively. $b(w)$ is an intercept independent of Q. A is a linear transformation which is the identity matrix in LMIR and LMC, and $[\alpha_1 I + \alpha_2 M_{tr}^\top]$ in TRLM and TAL. α_1, α_2, α_3, and α_4 are parameters.

The framework gives us some insight on matching questions: the key to calculating question similarity is how to represent data in CQA in a vector space, including words in questions, words in answers, and question categories. For LMIR and LMC, words and categories are represented in a heuristic way. The one-hot-spot representation is simple, but totally ignores the similarity of words. Different words are orthogonal in the vector space. Although there is a category vector in LMC, it is defined as a summation of one-hot-spot representations, and thus does not model the relations of words, either. TRLM and TAL leverage translation matrix M_{tr} and transform the one-hot-spot representation of word $v_{ohs}(w)$ to a new vector. Word similarity is encoded in the new vector. The problem with TRLM and TAL is that the estimation of M_{tr} needs a cubic time algorithm with respect to vocabulary size [10]. Therefore, it is difficult for TRLM and TAL to scale up to very large data.

The problems suffered by existing models motivate us to think of representing data in CQA in a better and more efficient way. In the next section, we will elaborate our method of learning representations for data in CQA and leveraging representations for question retrieval.

5. LEARNING REPRESENTATIONS OF DATA FOR QUESTION RETRIEVAL

Inspired by the framework in Section 4, we propose learning representations of data in CQA. The learning approach optimizes a question similarity function derived from the framework, models local relatedness of words, and can be implemented with an algorithm running in linear time with respect to number of words in training data. With the representations of data, we create a question retrieval by incorporating the representations into LMIR.

5.1 Learning Approach

We derive a question similarity function from Equation (5) and Equation (6). Specifically, we consider the following functions:

$$f(q, Q) = \prod_{w \in q} g(w, Q), \tag{7}$$

$$g(w, Q) = \frac{e^{y(w,Q)}}{\sum_{w' \in V} e^{y(w',Q)}}, \tag{8}$$

$$y(w, Q) = v(w) \cdot \left[\sum_{w' \in Q} v(w') + v\left(Cat(Q)\right) + \sum_{w' \in a} v(w') \right], \tag{9}$$

where V represents the vocabulary. From the equations above, we can see that $y(w, Q)$ is a special case of the general word-question similarity function given by Equation (6). Here, we set $\alpha_1 = \alpha_2 = \alpha_3 = 1$, fix $A = I$, and drop the intercept $b(w)$ in Equation (6). We fix A as an identity

matrix, because the information of the linear transformation can be encoded in the learning of vectors. The intercept $b(w)$ in the existing models is designed for avoiding zero matching scores due to orthogonality of word vectors. We try to learn a distributed representation that can reflect semantic relations of words. Therefore, we do not have to worry about the issue. In spite of this, one can still consider A and $b(w)$ in learning and see whether they can further improve retrieval performance. We leave this for future work. Differing from existing models, we apply the softmax function (i.e., $e^{y(w,Q)}/\sum_{w' \in V} e^{y(w',Q)}$) to $y(w, Q)$ and transform the linear combination of dot products to a probability distribution.

We expect to learn representations of words and question categories by maximizing the similarity between similar question pairs. The problem is that the learning needs large scale supervision signals indicating the similarity between question pairs which are not easy to obtain. Therefore, we consider maximizing the similarity between a question and itself. Our learning approach can be formulated as

$$\arg\max_{v(w), v(Cat(Q))} \prod_{i=1}^{N} f(Q_i, Q_i) = \prod_{i=1}^{N} \prod_{w \in Q_i} g(w, Q_i), \tag{10}$$

where $\{Q_i\}_{i=1}^N$ represents questions we have collected from CQA. We denote the category and the answer associated with Q_i by $Cat(Q_i)$ and a_i respectively. From the definition of $y(w, Q)$ in Equation (9), we know that by optimizing Equation (10), each word in a question is forced to be similar to every word in the question and every word in the answer. This is not reasonable, since in a question, especially in a long question, words appearing in a long distance are usually less related to each other. For example in a question like "what are good foods for a gymnast to eat", compared to "food", "what" is obviously less related to "gymnast". Since answers are usually much longer than questions, the problem could become more serious. The current definition of $y(w, Q)$ ignores locations of words and thus loses local structures within questions and answers. Besides this problem, in $y(w, Q)$, the category vector $v\,(Cat(Q))$ is equally weighted with each word vector $v(w')$ (see Equation (9)). Since we take a summation on word vectors, the effect of category vector is easily overwhelmed by the word vectors in learning. To solve these problems, we re-define the objective function in problem (10) by factorizing the probability $g(w, Q_i)$ in Equation (10). Each factor defines a probability distribution on words in which a word w is only similar to the category of Q_i and words in a small window around w in Q_i or a_i. Formally, the new objective function of our learning approach is given by:

$$\prod_{i=1}^{N} \prod_{w \in Q_i} \left[\prod_{j \in pos(w, Q_i)} \hat{g}(w, c(w, Q_i, j)) \prod_{j \in pos(w, a_i)} \hat{g}(w, c(w, a_i, j)) \right]. \tag{11}$$

$\forall j \in pos(w, Q_i)$,

$$\hat{g}(w, c(w, Q_i, j)) = \frac{e^{\hat{y}(w, c(w, Q_i, j))}}{\sum_{w' \in V} e^{\hat{y}(w', c(w, Q_i, j))}},$$

$$\hat{y}(w, c(w, Q_i, j)) = v(w) \cdot \left[\sum_{l=j-k, l \neq j}^{j+k} v(w_{(Q_i, l)}) + v(Cat(Q_i)) \right], \tag{12}$$

and $\forall j \in pos(w, a_i)$,

$$\hat{g}(w, c(w, a_i, j)) = \frac{e^{\hat{y}(w, c(w, a_i, j))}}{\sum_{w' \in V} e^{\hat{y}(w', c(w, a_i, j))}},$$

$$\hat{y}(w, c(w, a_i, j)) = v(w) \cdot \left[\sum_{l=j-k, l \neq j}^{j+k} v(w_{(a_i, l)}) + v(Cat(Q_i)) \right]. \tag{13}$$

In Equation (12) and Equation (13), $w_{(Q_i, l)}$ and $w_{(a_i, l)}$ represent the l-th word in question Q_i and answer a_i respectively. $pos(w, Q_i) = \{j | w_{(Q_i, j)} = w\}$ and $pos(w, a_i) = \{j | w_{(a_i, j)} = w\}$ indicate locations of word w in Q_i and a_i respectively. $c(w, Q_i, j)$ is defined as $\sum_{l=j-k, l \neq j}^{j+k} v(w_{(Q_i, l)}) + v(Cat(Q_i)$ which represents the context of word w in Q_i. $c(w, a_i, j)$ is similarly defined and represents the context of w in a_i. k is the window size. Note that in $\prod_{w \in Q_i}$ of Equation (11) we only enumerate distinct words in Q_i.

If we remove $\prod_{j \in pos(w, a_i)} \hat{g}(w, c(w, a_i, j))$ from Equation (11) and $v(Cat(Q_i))$ from Equation (12), then the learning method becomes

$$\arg\max_{v(w)} \prod_{i=1}^{N} \prod_{w \in Q_i} \prod_{j \in pos(w, Q_i)} \frac{e^{v(w) \cdot \sum_{l=j-k, l \neq j}^{j+k} v(w_{(Q_i, l)})}}{\sum_{w' \in V} e^{v(w') \cdot \sum_{l=j-k, l \neq j}^{j+k} v(w_{(Q_i, l)})}}.$$

This is equivalent to learning word representations using the continuous bag-of-words (CBOW) model [12, 15, 7] on question set $\{Q_i\}_{i=1}^N$. Our method extends CBOW by simultaneously embedding question categories and answer words.

Inspired by the connection of our method and CBOW, we consider optimizing the objective (11) by employing the negative sampling technique [12, 15, 7] which has proven both effective and efficient in learning word embeddings. Specifically, we learn the vectors of words and question categories by maximizing the following function:

$$\sum_{i=1}^{N} \sum_{w \in Q_i} [\sum_{j \in pos(w, Q_i)} \bar{g}(w, c(w, Q_i, j)) + \sum_{j \in pos(w, a_i)} \bar{g}(w, c(w, a_i, j))],$$

where

$$\bar{g}(w, c(x)) = \log \sigma(\hat{y}(w, c(x))) + \sum_{w' \in Neg(c(x))} \log[1 - \sigma(\hat{y}(w', c(x)))].$$

Here, $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid function. $\hat{y}(w, c(x))$ is defined by Equation (12) or Equation (13) depending on $c(x) = c(w, Q_i, j)$ or $c(x) = c(w, a_i, j)$. $Neg(c(x))$ represents a set of randomly sampled negative examples for $c(x)$. The optimization problem can be easily solved by stochastic gradient ascent. The gradient of $\bar{g}(w, c(x))$ is given by:

$$\frac{\partial \bar{g}(w, c(x))}{\partial v(Cat(Q))} = \sigma(-\hat{y}(w, c(x)))v(w) - h_{neg}(w, c(x)),$$

$\forall w' \in c(x)$:

$$\frac{\partial \bar{g}(w, c(x))}{\partial v(w')} = \sigma(-\hat{y}(w, c(x)))v(w) - h_{neg}(w, c(x)),$$

where $h_{neg}(w, c(x)) = \sum_{w' \in Neg(c(x))} \sigma(\hat{y}(w', c(x)))v(w')$, $\forall w' \in Neg(c(x))$:

$$\frac{\partial \bar{g}(w, c(x))}{\partial v(w')} = -\sigma(\hat{y}(w', c(x)))c(x),$$

and if w is the target word in $\bar{g}(w, c(x))$:

$$\frac{\partial \bar{g}(w, c(x))}{\partial v(w)} = \sigma(-\hat{y}(w, c(x)))c(x).$$

The training complexity is $O\left(M \times [2k \times D + D \times N_{neg}]\right)$, where M is the number of words in the training data, D is the dimension of vectors, and N_{neg} is the number of negative samples. The implementation code is published at `https://github.com/ComputerHobbyist/cqa`.

5.2 Retrieval Model

We consider creating a retrieval model with the learned representations of words and question categories. One straightforward way is to calculate $f(q, Q)$ and $g(w, Q)$ defined by Equation (7), Equation (8) and Equation (9) using the learned vectors and taking $f(q, Q)$ as a retrieval model. In practice, however, this strategy is infeasible, since there is a normalizer $\sum_{w' \in V} e^{y(w', Q)}$ in the definition of $g(w, Q)$, and for every candidate Q, we have to calculate its similarity with each word in the vocabulary. The time complexity is $O(|V|)$, and is too expensive for online retrieval and ranking. Moreover, from experimental results, we find that $f(q, Q)$ performs badly on measuring question similarity. Therefore, we propose an alternative way to leverage the learned representations for retrieval.

We borrow an idea from translation based language models, and consider incorporating word-word similarity and word-category similarity calculated with the learned vectors into LMIR. Specifically, suppose that there are a total of N_{cat} categories, and for the i-th category c_i, the vector representation is $v(c_i)$. We partition vocabulary V as $\{C_i\}_{i=1}^{N_{cat}}$ with $\{v(c_i)\}_{i=1}^{N_{cat}}$, and the i-th word cluster C_i is defined as $\{w_j \mid cosine\left(v(w_j), v(c_i)\right) > cosine\left(v(w_j), v(c_k)\right), \forall k \neq i\}$. In other words, we take the vector of categories as centroid of clusters, and assign each word to the cluster of its most similar category under Cosine similarity. Within each cluster, we find top N similar words for each word with Cosine similarity, and define a word-word similarity function as

$$P_{sim}(w_i|w_j) = \begin{cases} \frac{e^{v(w_i)\cdot v(w_j)}}{\sum_{w' \in Sim(w_j)} e^{v(w')\cdot v(w_j)}} & \text{if } w_i \in Sim(w_j) \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

Here, we use $Sim(w_j)$ to represent the top N similar words of word w_j. $P_{sim}(w_i|w_j)$ can be understand as a translation probability from w_j to w_i. Our idea is that we replace the translation probability $P_{tp}(w_i|w_j)$ in TRLM with $P_{sim}(w_i|w_j)$. The "new" TRLM is further combined with a word-category similarity function defined as

$$s_{cat}(w, c) = \frac{e^{v(w)\cdot v(c)}}{\sum_{w' \in V} e^{v(w')\cdot v(c)}}. \tag{15}$$

Thus, given a query question q and a candidate question Q, our retrieval model is given by

$$P(q|Q) = \prod_{w \in q} \left[(1-\lambda)P_{mx}(w|Q) + \lambda P_s(w|Q)\right], \tag{16}$$

where

$$\begin{aligned} P_{mx}(w|Q) &= (1-\alpha)P_{ml}(w|Q) + \alpha P_{sim}(w|Q), \\ P_s(w|Q) &= (1-\beta)P_{ml}(w|C) + \beta s_{cat}\left(w, Cat(Q)\right), \\ P_{sim}(w|Q) &= \sum_{v \in Q} P_{sim}(w|v)P_{ml}(v|Q). \end{aligned} \tag{17}$$

Table 1: Statistics of the two CQA data sets

	Yahoo	Baidu
Question Number	1,076,425	983,914
Description Number	872,981	552,959
Answer Number	7,567,268	3,768,391
Vocabulary Size	279,353	163,110

Here λ, α and β are parameters, satisfying λ, α, $\beta \in (0, 1)$. We name our retrieval model distributed representation based language model (DRLM).

DRLM has many advantages over translation based language models. First, it extends TRLM by considering category vectors, thus question matching is not only conducted on a word level, but also on a topic level. Second, it is more efficient than TRLM on model estimation. From Section 5.1, we know that the time complexity for learning representations of data is $O\left(M \times [2k \times D + D \times N_{neg}]\right)$. After learning, the time complexity for constructing the retrieval model is $O\left(|V| \times N_{cat} \times D + [\frac{|V|}{N_{cat}}]^2 \times D\right)$[1]. On the other hand, the time complexity for estimating a translation table for TRLM is $O\left(T \times |V|^2\right)$ in which T is the number of question-answer pairs [10]. In general, T is larger than $|V|$ on millions of data in CQA, thus in representing semantic relations of words, TRLM needs a cubic time algorithm with respect to vocabulary size to process data in CQA, while our method only needs to scan the data word by word a few times, and then we get both the representation of words and the representation of categories. Calculating similarity for pairs of words can be slow, but we only have to care about words in a small cluster. Moreover, since the vocabulary is partitioned, the estimation of P_{sim} can be implemented in parallel offline. In practice, we find that the estimation of DRLM is very efficient, and we can conclude that our method is easy to scale up and is capable of leveraging big data in CQA. Finally, word-word translation is calculated within clusters defined by category vectors. Therefore, besides the advantage on efficiency, our model can leverage categories to filter noise in similarity of words.

In DRLM, we only keep similarity of a word with its top N similar words, which makes the translation table very sparse. This strategy can not only further calibrate word similarity, but also keep online memory cost linearly increasing with the vocabulary size.

6. EXPERIMENTS

We conducted experiments to test the performance of the proposed method on question retrieval.

6.1 Experiment Setup

6.1.1 Data Sets

We crawled 1 million questions (a.k.a. question titles) associated with descriptions (a.k.a. question bodies) and answers from Yahoo! Answers and Baidu Knows. Yahoo! Answers and Baidu Knows represent the largest and most popular CQA archives in English and Chinese, respectively. In both Yahoo! Answers and Baidu Knows, there is a hierarchy of category. In Yahoo Answers, there are 26 root categories and 1263 leaf categories, while in Baidu Knows,

[1]In practice, we have observed that words almost uniformly distribute over clusters, and thus the size of each cluster is close to $|V|/N_{cat}$.

there are 14 root categories and 984 leaf categories. Each question in our crawled data has a root category and a leaf category (we ignore the categories on middle layers of the hierarchy). Table 1 gives details on the two data sets. We use the data sets for training models and have published them at `https://github.com/ComputerHobbyist/cqa`.

We test our model on two labeled sets published at `https://github.com/ComputerHobbyist/cqa`. One contains 1,685 English queries and each query has 15 labeled candidate questions. The other one contains 1,140 Chinese queries and each query has 13 labeled candidate questions. In both of the two sets, the candidates are returned by an inline ranking algorithm in Lucene.Net which is an open indexing source. The labels are either 1 or 0, meaning relevant or irrelevant. Note that the two labeled sets are expansions of those used in our previous work [28]. After the work, we further label 262 new English queries and 535 new Chinese queries and combine the newly labeled data with those in [28]. We follow [28] and randomly split each of the two data sets to a validation set and a test set with a ratio 1 : 3. On the validation set, we tune parameters of different models, while on the test set, we evaluate ranking performance of different models. For all English data, we remove stop words and conduct stemming, and for all Chinese data, we conduct Chinese word segmentation.

To evaluate the performance of different models, we follow [28], and employ Mean Average Precision (MAP) [1], Mean Reciprocal Rank (MRR) [23], R-Precision (R-Prec) [5], and precision at position 1 (P@1) as evaluation measures.

6.1.2 Baselines

We take the language model for information retrieval (LMIR) given by Equation (1) as a baseline method. In addition to that, we also consider the following baseline methods:

Language Model with Category Smoothing We implement LMC given by Equation (2). The word distribution under a category in LMC is estimated using the training data. Besides the leaf category smoothing technique used in [5], we also consider root categories of questions, and estimate a root category smoothing item. We denote a language model with leaf category smoothing as LMC_{leaf}, and a language model with root category smoothing as LMC_{root}.

Translation based Language Model We learn word-word translation probabilities from the training data using MGIZA++ [2] which is a multi-thread version of GIZA++. Since in [28], there is a comprehensive comparison between different variants of translation based language models, we select the best ones as baseline methods in this paper. For the English data set, we used $TAL_{Q\&D-AllA}$ in which a question is concatenated with its description as an enriched question, all answers under the question are concatenated as a long document, and the translation table is learned from pairs of the enriched question and the long answer document. For the Chinese data set, we use TAL_{Q-D} in which we learn the translation table from question-description pairs. For reference, we also report the comparison with TAL_{Q-D} and $TAL_{Q\&D-AllA}$ on the English data set and the Chinese data set, respectively.

In our model, we only keep top N similar words for each word in the translation table. To make a fair comparison, we also truncate the translation table in TAL by keeping top N words that can be translated from a word, and denote the models as TAL^{trunc}.

The original TAL in Equation (4) does not take categories into account. Therefore, we linearly combine $P_{mx}(w|Q,a)$ in Equation (4) with the leaf category smoothing item $P_{ml}(w|Cat(Q))$ in Equation (2), and incorporate the category information into TAL. The enriched TAL is denoted as TAL^{cat}.

LDA based Models We compare DRLM with four LDA based models in [28], including $TBLM_{LDA}$, $QATM + TransLM$, $TBLM_{QATM}$, and $TBLM_{SQATM}$ which is our major contribution in [28].

CBOW and Skip-gram We learn word vectors by employing the original CBOW and Skip-gram models with the negative sampling technique and use the learned word vectors to create several retrieval models for comparison. For both English data and Chinese data, we concatenate questions, descriptions, and answers as a document and run the algorithm of CBOW and Skip-gram using the code shared at `https://code.google.com/p/word2vec/`. With the word vectors, we create the following models: (1) we average word vectors in both a query question and a candidate question, and calculate the Cosine similarity of the two vectors. We denote the models as $CBOW_{cos}$ and $Skip_{cos}$; (2) We first find the top N similar words for each word using Cosine similarity, then calculate word-word translation probabilities using P_{sim}. We replace the translation probabilities in TRLM with the new probabilities, and denote the models as $CBOW_{trlm}$ and $Skip_{trlm}$. Note that this is equivalent to implementing our DRLM with word vectors learned using CBOW and Skip-gram. (3) We linearly combine $CBOW_{trlm}$ and $Skip_{trlm}$ with the leaf category smoothing item $P_{ml}(w|Cat(Q))$ in Equation (2) like what we did in TAL^{cat}. We denote the models as $CBOW^{cat}_{trlm}$ and $Skip^{cat}_{trlm}$.

In addition to DRLM, we also implement two simple models for comparison using the vectors learned by the method described in Section 5.1 . First, we average vectors of words in a question as a question vector, calculate Cosine similarity of two question vectors and Cosine similarity for a question vector and a category vector, and linearly combine the two cosine similarity functions. We denote this model as DR_{cos}. Second, we remove word-category similarity (i.e., $s_{cat}(w, Cat(Q))$ in Equation (17) from DRLM. We denote the model as $DRLM_{no-cat}$.

6.1.3 Parameter Tuning

We tune parameters for different models based on their performance on MAP. For LMIR, LMC and TAL, we selected parameters $\{\lambda, \alpha, \beta, \gamma\}$ from {0.1, 0.2, $\cdots$, 0.9}. For CBOW, Skip-gram, and our model, we set the number of negative samples (i.e.,N_{neg}) in {5, 10, 15}, the dimension of vectors (i.e.,D) in {100, 200, 300, 400, 500}, and the window size (i.e., k) in {1, 2, $\cdots$ 7}. In TAL^{trunc} and our model, we set N= 10,000 in the top N words. We tune the combination parameters in DRLM, DR_{cos}, TAL^{cat}, $CBOW_{trlm}$, $Skip_{trlm}$, $CBOW^{cat}_{trlm}$, and $Skip^{cat}_{trlm}$ in the same way as in LMIR, LMC, and TAL.

6.2 Evaluation Results

We run algorithms on a workstation with one Intel Core i7 4785T processor (4 cores). For our method, we use root categories for embedding, and concatenate answers under a

[2] `http://www.kyloo.net/software/doku.php/mgiza:overview`

Table 2: Evaluation results on Yahoo data and Baidu data (1 million)

	Yahoo data				Baidu data			
	MAP	MRR	R-Prec	P@1	MAP	MRR	R-Prec	P@1
$LMIR$	0.758	0.844	0.661	0.759	0.724	0.787	0.610	0.681
LMC_{root}	0.771	0.860	0.684	0.781	0.734	0.806	0.626	0.701
LMC_{leaf}	0.773	0.863	0.685	0.783	0.737	0.805	0.632	0.703
TAL_{Q-D}	0.770	0.856	0.681	0.776	0.735	0.807	0.631	0.704
$TAL_{Q\&D-AllA}$	0.774	0.863	0.687	0.784	0.727	0.794	0.618	0.694
TAL^{cat}	0.776	0.864	0.689	0.788	0.743	0.811	0.638	0.709
TAL^{trunc}	0.777	0.862	0.688	0.785	0.739	0.808	0.636	0.707
$QATM + TransLM$	0.769	0.856	0.683	0.774	0.733	0.804	0.627	0.704
$TBLM_{LDA}$	0.772	0.860	0.683	0.782	0.737	0.804	0.632	0.705
$TBLM_{QATM}$	0.773	0.868	0.686	0.786	0.730	0.799	0.620	0.696
$CBOW_{cos}$	0.719	0.810	0.618	0.710	0.682	0.757	0.556	0.649
$Skip_{cos}$	0.719	0.821	0.614	0.715	0.686	0.765	0.561	0.660
$CBOW_{trlm}$	0.772	0.859	0.686	0.781	0.741	0.812	0.632	0.712
$Skip_{trlm}$	0.773	0.861	0.684	0.785	0.743	0.813	0.633	0.715
$CBOW^{cat}_{trlm}$	0.773	0.863	0.685	0.787	0.745	0.815	0.636	0.718
$Skip^{cat}_{trlm}$	0.775	0.866	0.689	0.792	0.747	0.818	0.642	0.721
DR_{cos}	0.722	0.824	0.624	0.720	0.690	0.771	0.570	0.669
$DRLM_{no-cat}$	0.774	0.861	0.687	0.786	0.745	0.815	0.637	0.719
$TBLM_{SQATM}$	0.780	0.869	0.694	0.793	0.741	0.814	0.640	0.710
$DRLM$	**0.789**	**0.881**	0.701	**0.816**	**0.761**	**0.835**	**0.657**	**0.738**

question as a long answer document. Table 2 lists evaluation results of different models on the two data sets.

From Table 2, we can see that DRLM outperforms all the baseline methods on both Yahoo data and Baidu data, and the improvements, except on R-Prec on Yahoo data over $TBLM_{SQATM}$, are statistically significant over all the baseline methods on all metrics (t-test, p value < 0.05).

DRLM significantly outperforms LMIR and LMC, which demonstrates that the learned representations in DRLM can express semantic relations of words and effectively bridge the lexical gaps in similar questions. It significantly improves upon the CBOW and Skip-gram based methods. Particularly, it is much better than the heuristic combination of embedding based translation models (i.e., $CBOW_{trlm}$ and $Skip_{trlm}$) and the category smoothing item (i.e., $P_{ml}(w|Cat(Q))$. This result indicates that category embedding can reinforce the effect of word embedding, and word-category similarity calculated in the vector space is very useful for identifying the semantic similarities between questions. $DRLM_{no-cat}$ is comparable with $CBOW_{trlm}$ and $Skip_{trlm}$, which further emphasizes the importance of category embedding in our model. DR_{cos} performs much worse than DRLM, which tells us that the trivial method of using the learned representations does not work in retrieval. DRLM outperforms all LDA based models. Particularly, it outperforms $TBLM_{SQATM}$ even though it does not leverage information on answer quality. The results indicate that the embedding approach we proposed is more effective at capturing semantic relations of words and categories. Finally, we find that DRLM is much better than translation based language models on both efficacy and efficiency. Particularly, on efficiency, although we have exploited a multi-threaded tool, it still took us more than 2 days to learn the translation table for TAL. On the other hand, with the same data and on the same machine, our method only needs 9 minutes to finish the learning of data representations and less than 40 minutes to estimate the retrieval model after learning.

LMC_{leaf} is slightly better than LMC_{root}, indicating that word distributions under fine-grained categories are more useful for question retrieval. Truncation on the translation table of TAL only slightly improves performance. We find that many translation probabilities are already zero in the original translation table, as many word pairs scarcely co-occurred in question-answer pairs. Skip-gram based methods are slightly better than CBOW based methods. The result is consistent with the conclusion drawn by Mikolov et al. in [15]. $Skip_{cos}$ and $CBOW_{cos}$ perform badly, indicating that the average of word vectors cannot represent the semantics of questions. The bad performance of DR_{cos} also verified this conclusion.

6.3 Discussions

Experiments on 1 million data have verified the advantages of our method on both efficacy and efficiency. In fact, the high efficiency of DRLM allows us to learn representations with bigger data. Therefore, we collect an extra 9 million questions associated with descriptions and answers from both Yahoo! Answers and Baidu Knows. Together with the 1 million data from before, we have a total of 10 million data for both the Yahoo data set and the Baidu data set. We conduct experiments on the bigger data sets, and study (1) if DRLM is still better than the baseline methods; (2) how the efficacy and efficiency of DRLM change with the growth of data.

In addition to the experiments on the bigger data sets, we use a real example to explain how the learned representations of words and question categories in DRLM help bridge lexical gaps between similar questions. We also compare DRLM with $f(q, Q)$ given by Equation (7), in which $f(q, Q)$ is calculated using vectors learned by the method in Section 5.1.

6.3.1 Comparison of retrieval models on bigger data

Table 3 gives the evaluation results of different models on 10 million data. On such a scale, TAL and $TBLM_{SQATM}$ become infeasible (e.g., taking months to run), therefore, we only compare DRLM with LMIR, the best LMC (i.e., LMC_{leaf}), and the best embedding based baseline (i.e., $Skip^{cat}_{trlm}$). From the table, we can see that (1) after we

increase training data, all methods become better; (2) Our method still significantly outperforms all baseline methods on all metrics (t-test, p value < 0.05), which is consistent with the results on the 1 million pieces of data;

6.3.2 Changes of DRLM with the growth of data

Besides comparisons between different models, we also examine how the efficacy and efficiency of DRLM changes with the growth of data. Specifically, we train DRLM on 1 million, 3 million, 5 million, 7 million and 10 million data. Figure 1 shows how the efficacy of DRLM changes with the growth of data. We can see that the upward trend on P@1 is very clear. DRLM becomes better and better on P@1 when training data increases. On other metrics, although the performance of DRLM drops sometimes, the overall trend is similar to that on P@1. Figure 2 shows how the training time changed with the growth of data. We can see that the training time almost linearly increases with training data[3], which demonstrates the scalability of our method. Particularly, on 10 million data, our method only needs 50 minutes to learn the representations of data, and less than 200 minutes to estimate the retrieval model with the representations, which is very efficient.

6.3.3 Case study

We try to explain why DRLM is effective in measuring question similarity with a real example from the test data. For the query "how to cut bicycle shifter cable", there is a relevant candidate "how do I remove cable housing from a shifter on a bike". The two questions express similar meanings with different but semantically related words such as "bike" and "bicycle" and "cut" and "remove". DRLM ranks the candidate in the top position. We check the learned vectors and find that the vectors of "bike" and "bicycle" and the vectors of "cut" and "remove" are grouped into the same clusters. Given "bike" and " remove", the words that correspond to the largest translation probabilities (i.e., $P_{sim}(w_i|w_j)$ in Equation (14)) are "bicycle" and "cut", respectively. Moreover, the category of the candidate is "Sports" and the similarity between the vector of "bicycle" in the query and the vector of "Sports" is very large. Through learning, our method embeds words and categories into a vector space where semantically similar ones are close to each other. Therefore, DRLM can successfully capture the similarity between the query and the candidate even though they share few common words. On the other hand, LMIR fails on this case. It ranks the relevant candidate in a lower position than another irrelevant candidate "how do I find out what type of bicycle cables I need in order to replace them on my bike" due to the more common words shared by the query and the irrelevant candidate.

6.3.4 Comparison of $f(q, Q)$ and DRLM

As we have explained in Section 5, we do not employ $f(q, Q)$ defined by Equation (7) as a retrieval model because the normalizer in $g(w, Q)$ cannot be efficiently calculated. In spite of this, we still want to study if $f(q, Q)$, as a similarity measure of question pairs, is better than DRLM when we calculate $f(q, Q)$ using the vectors learned by the method described in Section 5.1.

Table 4 shows the comparison of the two models. We can see that $f(q, Q)$ is much worse than DRLM. From the results, we can conclude that the retrieval model we propose in Section 5.2 is superior in both efficiency and efficacy on measuring question similarity.

[3]The number of words does not increase as fast as the number of training data.

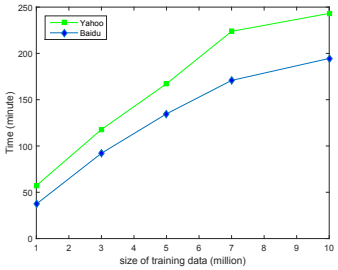


Figure 2: Changes of training time (mins) of DRLM on Yahoo data and Baidu data

7. CONCLUSION

We propose learning distributed representations of data in CQA and leveraging the representations for question retrieval. The learning approach is inspired by a new framework for existing retrieval models. Our approach simultaneously embeds words and categories of questions into a vector space and models local relatedness of words in learning. We create a retrieval model with the learned representations. Evaluation results on public data sets show that the proposed method can significantly improve upon existing models in terms of both efficacy and efficiency.

8. ACKNOWLEDGEMENT

The work is supported in part by the National Natural Science Foundation of China (Grant Nos. 61170189, 61370126, 61202239), National High Technology Research and Development Program of China under grant (No.2015AA016004), Major Projects of the National Social Science Fund of China under grant (No.14&ZH0036), Science and Technology Innovation Ability Promotion Project of Beijing (PXM2015-014203-000059), the Fund of the State Key Laboratory of Software Development Environment (No. SKLSDE-2015ZX-16), and Microsoft Research Asia Fund (No. FY14-RES-OPP-105).

9. REFERENCES

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *JMLR '03*, 3:1137–1155, Mar. 2003.

[3] A. Berger, R. Caruana, D. Cohn, D. Freitag, and V. Mittal. Bridging the lexical chasm: statistical approaches to answer-finding. In *SIGIR'00*, pages 192–199, 2000.

[4] X. Cao, G. Cong, B. Cui, and C. S. Jensen. A generalized framework of exploring category information for question retrieval in community question answer archives. In *WWW'10*, pages 201–210, 2010.

[5] X. Cao, G. Cong, B. Cui, C. S. Jensen, and C. Zhang. The use of categorization information in language models for question retrieval. In *CIKM'09*, pages 265–274, 2009.

[6] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML '08*, pages 160–167, New York, NY, USA, 2008. ACM.

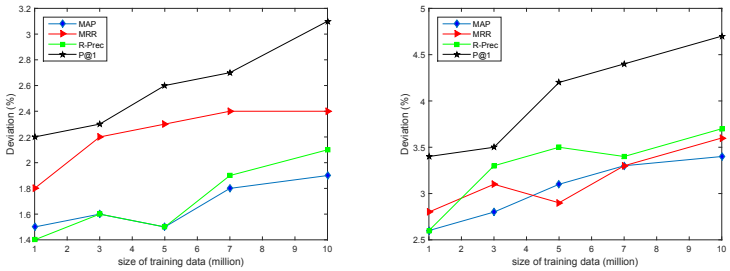
[7] Y. Goldberg and O. Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

Table 3: Evaluation results on Yahoo data and Baidu data (10 million)

	Yahoo data				Baidu data			
	MAP	MRR	R-Prec	P@1	MAP	MRR	R-Prec	P@1
$LMIR$	0.765	0.850	0.675	0.763	0.740	0.810	0.631	0.707
LMC_{leaf}	0.778	0.868	0.691	0.794	0.747	0.816	0.637	0.718
$Skip^{cat}_{trlm}$	0.782	0.874	0.693	0.802	0.756	0.827	0.654	0.737
$DRLM$	**0.794**	**0.887**	**0.708**	**0.825**	**0.769**	**0.843**	**0.668**	**0.751**

Table 4: Comparison of $f(q, Q)$ and DRLM on 1 million data

	Yahoo data				Baidu data			
	MAP	MRR	R-Prec	P@1	MAP	MRR	R-Prec	P@1
$f(q, Q)$	0.654	0.738	0.554	0.606	0.605	0.667	0.470	0.516
$DRLM$	0.789	0.881	0.701	0.816	0.761	0.835	0.657	0.738



(a) Changes of efficacy of DRLM on Yahoo data (b) Changes of efficacy of DRLM on Baidu data

Figure 1: Changes of efficacy of DRLM on the two data sets. y axis represents the percentage of deviation of DRLM from TAL trained using 1 million data.

[8] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. Improving word representations via global context and multiple word prototypes. In *ACL'12*, pages 873–882, Stroudsburg, PA, USA, 2012. ACL.

[9] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *CIKM '13*, pages 2333–2338, New York, NY, USA, 2013. ACM.

[10] J. Jeon, W. B. Croft, and J. H. Lee. Finding similar questions in large question and answer archives. In *CIKM'05*, pages 84–90, 2005.

[11] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *ICML '14*, pages 1188–1196, 2014.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[13] T. Mikolov, M. KarafiÃąt, L. Burget, J. CernockÃ¡, and S. Khudanpur. Recurrent neural network based language model. In T. Kobayashi, K. Hirose, and S. Nakamura, editors, *INTERSPEECH*, pages 1045–1048. ISCA, 2010.

[14] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168, 2013.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS '13*, pages 3111–3119. Curran Associates, Inc., 2013.

[16] A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In *NIPS '09*, pages 1081–1088. Curran Associates, Inc., 2009.

[17] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In R. G. Cowell and Z. Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252. Society for Artificial Intelligence and Statistics, 2005.

[18] J. Ponte and W. Croft. A language modeling approach to information retrieval. In *SIGIR' 98*, pages 275–281, 1998.

[19] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *TREC*, 1994.

[20] H. Schwenk, D. Dchelotte, and J.-L. Gauvain. Continuous space language models for statistical machine translation. In *COLING-ACL '06*, pages 723–730, Stroudsburg, PA, USA, 2006. ACL.

[21] H. Schwenk and J.-L. Gauvain. Neural network language models for conversational speech recognition. In *International Conference on Speech and Language Processing*, pages 1215–1218, 2004.

[22] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *CIKM '14*, pages 101–110, New York, NY, USA, 2014. ACM.

[23] E. M. Voorhees. The trec-8 question answering track report. In *Proceedings of the 8th Text Retrieval Conference*, pages 77–82, 1999.

[24] K. Wang, Z. Ming, and T.-S. Chua. A syntactic tree matching approach to finding similar questions in community-based qa services. In *SIGIR'09*, pages 187–194, 2009.

[25] J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI'11*, pages 2764–2770. AAAI Press, 2011.

[26] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *SIGIR'08*, pages 475–482, 2008.

[27] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.

[28] K. Zhang, W. Wu, H. Wu, Z. Li, and M. Zhou. Question retrieval with high quality answers in community question answering. In *CIKM '14*, pages 371–380, New York, NY, USA, 2014. ACM.

[29] G. Zhou, L. Cai, J. Zhao, and K. Liu. Phrase-based translation model for question retrieval in community question answer archives. In *ACL'11*, pages 653–662, 2011.