Learning a Dual-Language Vector Space for Domain-Specific Cross-Lingual Question Retrieval

Guibin Chen[1], Chunyang Chen[1], Zhenchang Xing[1], and Bowen Xu[2]
[1]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[2]College of Computer Science and Technology, Zhejiang University, China
{gbchen, chen0966, zcxing}@ntu.edu.sg, max_xbw@zju.edu.cn

ABSTRACT

The lingual barrier limits the ability of millions of non-English speaking developers to make effective use of the tremendous knowledge in Stack Overflow, which is archived in English. For cross-lingual question retrieval, one may use translation-based methods that first translate the non-English queries into English and then perform monolingual question retrieval in English. However, translation-based methods suffer from semantic deviation due to inappropriate translation, especially for domain-specific terms, and lexical gap between queries and questions that share few words in common. To overcome the above issues, we propose a novel cross-lingual question retrieval based on word embeddings and convolutional neural network (CNN) which are the state-of-the-art deep learning techniques to capture word- and sentence-level semantics. The CNN model is trained with large amounts of examples from Stack Overflow duplicate questions and their corresponding translation by machine, which guides the CNN to learn to capture informative word and sentence features to recognize and quantify semantic similarity in the presence of semantic deviations and lexical gaps. A uniqueness of our approach is that the trained CNN can map documents in two languages (e.g., Chinese queries and English questions) in a dual-language vector space, and thus reduce the cross-lingual question retrieval problem to a simple k-nearest neighbors search problem in the dual-language vector space, where no query or question translation is required. Our evaluation shows that our approach significantly outperforms the translation-based method, and can be extended to dual-language documents retrieval from different sources.

CCS Concepts

•Software and its engineering → Software libraries and repositories; •Information systems → Multilingual and cross-lingual retrieval;

ASE'16, September 3–7, 2016, Singapore, Singapore
© 2016 ACM. 978-1-4503-3845-5/16/09...$15.00
http://dx.doi.org/10.1145/2970276.2970317

Keywords

Cross-lingual question retrieval, Word embeddings, Convolutional Neural Network, Dual-Language Vector Space

1. INTRODUCTION

Question Answering (Q&A) sites have become an important service for knowledge sharing and acquisition. In the software engineering domain, Stack Overflow is the most prominent Q&A site. Over the past few years, it has accumulated a large amount of user-generated content, which makes it a valuable repository of software engineering knowledge. The content in the Q&A sites is organized as questions and corresponding answers. One key task for reusing content in such a site is finding questions that are similar to user queries, as questions are the keys to accessing the knowledge in the site.

Many techniques [24, 28, 34] support *monolingual question retrieval*. That is, to retrieve English questions[1] in Stack Overflow, user queries must also be in English. However, many developers are from non-English speaking countries, such as China. According to Stack Overflow user and visit statistics, the U.S has 80 times more registered users than the China on Stack Overflow,[2] but the monthly visits from the U.S is only about 22 times more than that from the China[3]. This indicates that although not many Chinese developers participate in Q&As, much more of them do visit Stack Overflow to reuse its content. The English skills of non-English speaking developers could be good enough to read posts in English, but are usually not sufficient to express their questions in English queries. This limits the ability of non-English speaking developers to make effective use of Stack Overflow content.

We thus need *cross-lingual question retrieval* (CLQR) technique that overcomes the language barrier, and allows non-English speaking developers to issue queries in their native language (e.g., Chinese) to retrieve questions in English. One possible way for cross-lingual question retrieval to work is to first translate the queries from the native language into English, and then to use monolingual question retrieval to retrieve questions in English. This approach assumes that query translation can preserve query semantics, and question retrieval techniques can handle lexical gaps between

[1]Several non-English Stack Overflow sites have been launched. However, they are much less popular than English Stack Overflow.
[2]http://nlpx.net/archives/172
[3]http://stackoverflow.com/research/developer-survey-2016#overview

queries and questions. We argue that these assumptions are too simplistic.

Semantic deviation is likely to be introduced in query translation, especially for domain-specific technical terms. For example, "审查" has several common translations like "examine", "censor", or "investigate" by Google Translate. However, when we consider it in a specific sentence within software-engineering context, it would be better translated into some domain-specific terms in English. For example, for the two Chinese queries, "代码**审查**工具" and "网页元素**审查**", the appropriate domain-specific translations would be "code **review** tool" and "web element **inspection**". That is, "审查" should be translated as "review" and "inspections" in these two queries, while Google Translate cannot make the appropriate domain-specific translation. Although the English words "examine", "review" and "inspection" are synonyms, an inappropriate translation will introduce deviation from the original meanings of the queries and affect the subsequent retrieval step.

In addition to term-level semantic deviation across languages, sentence-level lexical gaps could further affect the performance of the retrieval techniques. Lexical gap means that relevant queries and questions may not share many words in common. As there are many ways to ask the same question, a user might not be able to find the answer if they are asking it a different way. Assume developers want to know how to read a text file. Some may describe their needs using the query "读取文本文件" ("read text file"), while others may use queries like "ASCII文件到字符串" ("ASCII file to string"), or "访问文件字符串内容" ("access string content in file"). Developers may also express errors of their program as queries like "不能加载全部字符串" ("cannot load all strings"). Stack Overflow already has good answers to the question like "read a plain text file". Although all the queries and the question share very similar meanings, except the query "read text file", the other queries and the question share few words in common. Such sentence-level lexical gap has become a major barricade for traditional information retrieval (IR) models (e.g., BM25 [28], LDA [2]) to determine query-question similarity [46]. For cross-lingual retrieval, the issue would be more prevalent, due to different norms and expressions across languages and semantic deviation in query translation.

In this paper, we tackle the cross-lingual question retrieval problem using deep learning techniques. To overcome the above issues on semantic deviation and lexical gap, we adopt word embeddings and Convolutional Neural Networks (CNN) to recognize and quantify word- and sentence-level semantic similarity across lingual barrier and lexical gap. We focus on Chinese-to-English question retrieval, because Chinese is distant from English, making it a more challenging task. The key innovation of our approach is that we train the CNN using sufficient examples of sentence pairs that are semantically equivalent but have semantic deviations due to inaccurate translation and lexical gaps. These training examples are collected from the large amount of duplicate questions (0.3 million) in Stack Overflow and the corresponding Chinese translations of these questions translated by machine (Google Translate). We design effective loss functions to guide the CNN to learn to capture the most informative word and sentence features to determine semantic similarity in the presence of semantic deviations and lexical gaps.

After training, our CNN can map both English and Chinese documents onto a *dual-language vector space* in which terms and documents from both languages are represented as language-independent vectors. Semantically close terms and documents from both languages are likewise close in the dual-language vector space. In this work, the English documents are a knowledge base of English questions, and the Chinese documents are Chinese queries. In contrast to query translation followed by monolingual question retrieval, our approach directly quantifies semantic similarity between Chinese queries and English questions in terms of their distance in the dual-language vector space in which no query or question translation is required. Given a query, cross-lingual question retrieval is then transfered to a simple k-nearest neighbors search in the dual-language vector space.

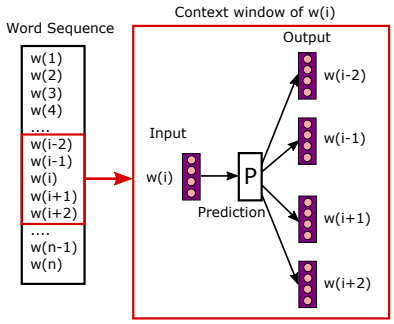


Figure 1: Continuous Skip-gram model

We compare our deep-learning based approach with the baseline GoogleTranslate+Lucene method. The results show that our approach obtains slightly better results than the baseline method in terms of the rank of the first relevant question retrieved. However, our approach significantly outperforms the baseline method in retrieving more relevant questions, especially the questions that exhibit a lexical gap with the input queries, which the baseline method usually fails to retrieve. Furthermore, our approach is more robust to the semantic deviations across languages. We also apply our CNN to a completely different data source (English version and Chinese version of a Python tutorial website), which demonstrate the generality of our approach.

The remainder of the paper is organized as follows. Section 2 introduces background related to word embeddings and CNN. Section 3 presents the technical details of our approach. Section 4 reports the evaluation of our approach. Section 5 reviews the related work. Section 6 summarizes our work and outlines the future plans.

2. BACKGROUND

This section introduces the basic concepts of the two key techniques, i.e., word embeddings and convolutional neural network, that our approach relies on.

2.1 Word Embeddings

Word embeddings are dense low-dimensional vector representations of words that are build on the assumption that words with similar meanings tend to be present in similar context. Studies [1, 7, 19, 30] show that word embeddings are able to capture rich semantic and syntactic properties of words, compared with one-hot word representation [32].

Word embeddings are typically induced using neural language model, which uses neural networks as the underlying predictive model. Figure 1 shows the continuous Skip-gram model [19], one of the two popular word-to-vector

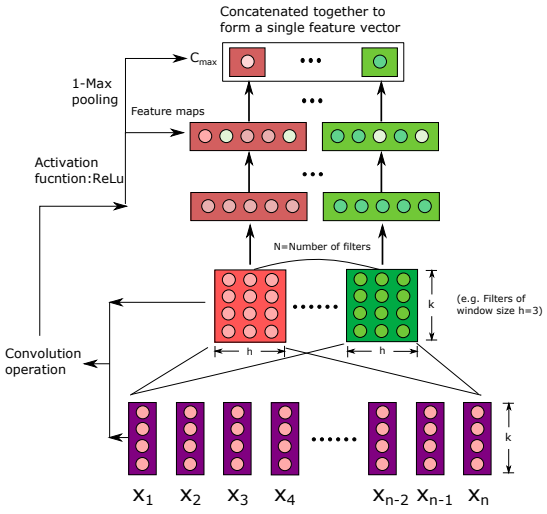


Figure 2: The architecture of a simple CNN

(word2vec) neural language models proposed by Mikolov et al. The goal of the continuous Skip-gram model is to learn the word embeddings of a center word (i.e., w_i) that is good at predicting the surrounding words in a context window of $2t+1$ words ($t=2$ in this example). More specifically, the objective function of the Skip-gram model is to maximize the sum of log probabilities of the surrounding context words conditioned on the center word:

$$\sum_{i=1}^{n} \sum_{-t \le j \le t, j \neq 0} \log p\left(w_{i+j}|w_i\right)$$

where w_i denotes the center word in a context window of length $2t+1$ and w_{i+j} denotes the context word surrounding w_i within the context window. n denotes the length of the word sequence. The $\log p\left(w_{i+j}|w_i\right)$ is the conditional probability defined using the *softmax function*:

$$p\left(w_{i+j}|w_i\right) = \frac{\exp(v'^{\mathrm{T}}_{w_{i+j}} v_{w_i})}{\sum_{w \in W} \exp(v'^{\mathrm{T}}_{w} v_{w_i})}$$

where v_w and v'_w are respectively the input and output vectors of a word w in the underlying a neural network, and W is the vocabulary of all words. Intuitively, $p\left(w_{i+j}|w_i\right)$ estimates the normalized probability of a word w_{i+j} appearing in the context of a center word w_i over all words in the vocabulary. This probability can be efficiently estimated by the negative sampling method [20].

The continuous Skip-gram model does not care about the input language as long as the sentences can be properly tokenized into a sequence of words. Given word sequences, the model maps words onto a low-dimensional, real-valued vector space. Word vectors are essentially feature extractors that encode semantic and syntactic features of words in their dimensions. In this vector space, semantically close words are likewise close in Euclidean distance.

2.2 Convolutional Neural Network

A convolutional neural network (CNN) utilizes layers with convolution filters that are applied to local features of an object (e.g., an image or a sentence) to produce a feature vector of the object [17]. Some recent works have successfully applied CNNs to model sentence- and document-level semantics for NLP tasks such as sentence classification [44], and duplicate question detection [3]. Figure 2 shows the model architecture of a simple CNN for NLP tasks. To apply the CNN to text, words comprising a sentence need to be converted into vector representations to be used as input to the CNN. A commonly used word vector representation is word embeddings as mentioned in the previous section. Let $x_i \in \mathbb{R}^k$ be the k-dimensional word vector corresponding to the i-th word in the sentence. A sentence of length n is represented as

$$x_{1:n} = x_1 \oplus x_2 \oplus ... \oplus x_n$$

where $\oplus$ is the vector concatenation operator. We can treat the sentence vector as an "image", and perform convolution on it via linear filters. In text applications, because each word is represented as a k-dimensional vector, it is reasonable to use filters with widths equal to the dimensionality of the word vectors (i.e., k). Thus we simply vary the *window size* (or *height*) of the filter, i.e., the number of adjacent words considered jointly. Let $x_{i:i+h-1}$ refer to the concatenation vector of h adjacent words $x_i, x_{i+1}, ..., x_{i+h-1}$. A convolution operation involves a *filter* $w \in \mathbb{R}^{hk}$ (a vector of $h \times k$ dimensions) and a bias term $b \in \mathbb{R}^h$, which is applied to h words to produce a new value $o_i \in \mathbb{R}$:

$$o_i = w^T \cdot x_{i:i+h-1} + b$$

where $i = 1...n-h+1$, and $\cdot$ is the dot product between the filter vector and the word vector. This filter is applied repeatedly to each possible window of h words in the sentence (i.e., $x_{1:h}, x_{2:h+1}, ..., x_{n-h+1:n}$) to produce an *output sequence* $o \in \mathbb{R}^{n-h+1}$, i.e., $o = [o_1, o_2, ..., o_{n-h+1}]$. We apply a non-linear *activation function* f to each o_i to produce a *feature map* $c \in \mathbb{R}^{n-h+1}$ where $c_i = f(o_i)$. A commonly used non-linear activation function is $ReLu$:

$$ReLu(o_i) = max(0, o_i)$$

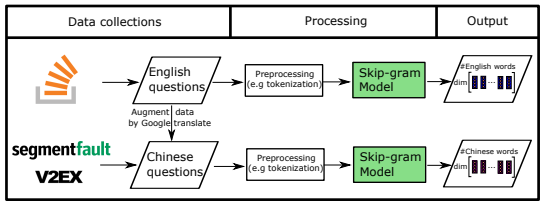
One may also specify multiple kinds of filters with different window sizes, or use multiple filters for the same window size to learn complementary features from the same word windows. In Figure 2, we illustrate N filters for the window size of $h=3$. The dimensionality of the feature map generated by each filter will vary as a function of the sentence length and the filter's window size. Thus, a pooling function is then applied to each feature map to induce a fixed-length vector. A common strategy is *1-max pooling*, which extracts a scalar (i.e., a feature vector of length 1) with the maximum value for each filter. Together, the outputs from each filter can be concatenated into a feature vector for one layer of the CNN. This feature vector can be fed into the next layer of the CNN for further convolution, or be used as the output vector for different NLP tasks (e.g, sentence classification [44], duplication question detection [3]).

3. DUAL-CHANNEL CNN FOR CLQR

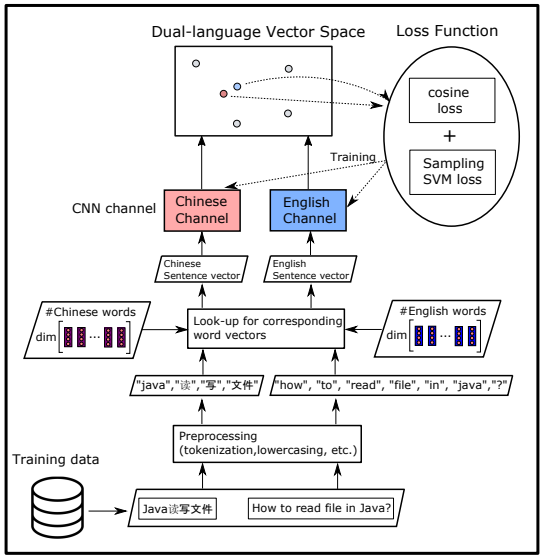
We now describe our dual-channel CNN that is designed for cross-lingual question retrieval (CLQR).

3.1 Approach Overview

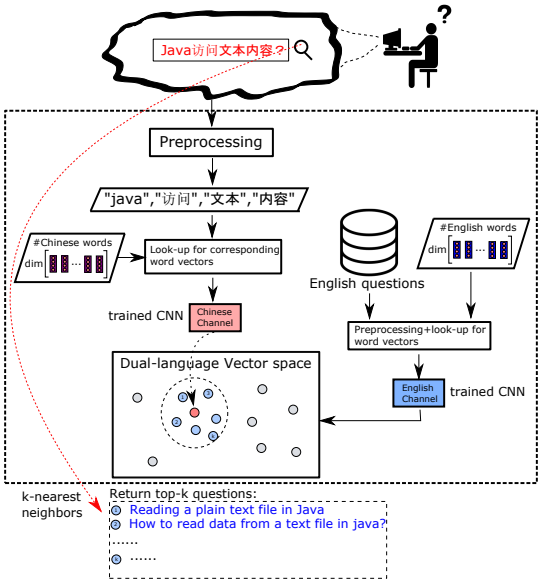
This work focuses on Chinese-to-English question retrieval, because Chinese is distant from English. Figure 3 presents the main steps of our approach. First, we collect questions from Stack Overflow and two Chinese Q&A sites (SegmentFault and V2EX) as our dataset. To mitigate the lack of



(a) Learning monolingual word embeddings



(b) Training the CNN



(c) Cross-lingual question retrieval

Figure 3: Overview of Main Steps

Chinese questions for learning Chinese word embeddings, we augment the Chinese question dataset with the Chinese translations of Stack Overflow questions using Google Translate. We learn monolingual word embeddings from a large number of English and Chinese questions, respectively. These pre-trained English and Chinese word embeddings are used in two ways: first as word vector representations of English and Chinese sentences for training the CNN, and second, as word vector representations of Chinese queries and English questions for cross-lingual retrieval.

Word embeddings capture word-level semantics for each language. Next, we use CNNs to quantify sentence-level semantic similarity across language. The application of CNNs includes a training phase and a query phase. In the training phase, we generate a set of dual-language sentence pairs, i.e., one sentence in English and the other in Chinese, from Stack Overflow duplicate questions and their corresponding Chinese translations (also obtained by Google Translate). A dual-channel CNN is designed, one channel for each language respectively, and trained by the dual-language sentence pairs. In the query phase, the respective CNN channel can map Chinese queries and a knowledge base of English questions onto the dual-language vector space in which terms and sentences from both languages are represented as language-independent vectors. In this vector space, semantically close terms and sentences from both languages are likewise close (see examples in Figure 7a). Given a query in Chinese, cross-language question retrieval is reduced to find k-nearest English questions close to the query in the dual-language vector space.

It is important to note that we use Google Translate to obtain Chinese translations of Stack Overflow questions for learning Chinese word embeddings and training the dual-channel CNN. However, cross-language question retrieval is supported by the dual-channel CNN in which no query or question translation is required.

3.2 Learning Monolingual Word Embeddings

Words are discrete symbols and cannot be fed directly to a neural network. We need to map each word to a real-valued vector. In this work, we use word embeddings as input word vector representations, because of the ability of word embeddings to capture latent semantic and syntactic features of words in their dimensions [19, 20].

To learn word embeddings of a language, we need large amounts of texts in that language. For software engineering question retrieval, texts should be domain-specific and cover the vocabulary of the questions people may ask. Therefore, we collect question titles from Stack Overflow as a corpus of English software engineering texts for learning English word embeddings. For learning Chinese word embeddings, we collect question titles from two Chinese Q&A sites (SegmentFault and V2EX) as a corpus of Chinese software engineering texts. Because Chinese Q&A sites have many fewer questions than Stack Overflow, we augment Chinese software engineering texts with the Chinese translations of Stack Overflow question titles. Due to the sheer amount of texts needed, human translation is impractical. Therefore, we use machine translation (Google Translate) to obtain Chinese translations of Stack Overflow question titles. Machine translation also "intentionally" injects some semantic deviations in between English and Chinese texts so that the trained CNN can be more robust to semantic deviations between Chinese queries and English questions.

Each collected question title is considered as a sentence. We preprocess English sentences by standard English text preprocessing steps like tokenization, lowercasing, etc. For Chinese sentences, we tokenize them into Chinese words using SnowNLP tool[4], and lowercase any English words (e.g., tool names) in the Chinese sentences. We use the continuous skip-gram model [20] (the Python implementation in

[4]https://github.com/isnowfy/snownlp

Gensim [26]) to learn monolingual word embeddings from the English and Chinese software engineering text, respectively. The output is a dictionary of English words and their embeddings and a dictionary of Chinese words and their embeddings, which will be used to represent English questions and Chinese queries respectively.

3.3 Training the Dual-Channel CNN

This section describes the architecture of our CNN, and how we generate training documents and train the CNN with these documents.

3.3.1 The Architecture of the Proposed CNN

For cross-lingual retrieval, we design a dual-channel CNN, each channel can map sentences in one language into a dual-language vector space in which sentence-level semantic similarity can be quantified. Both channels have the same architecture as shown in Figure 4. The difference is only that one takes English sentences as input while the other takes Chinese sentences as input. An input sentence is first converted into a sentence vector by looking up the word-embeddings dictionary and concatenating the word embeddings of the words comprising the sentence. The sentence vector is then fed into the CNN as input.

Our CNN consists of two layers, each of which follows the architecture depicted in Figure 2. The first layer of the CNN uses filters with three different window sizes (i.e., $h = 1,\ 3,\ 5$). That is, we try to extract features of 1-gram, 3-grams and 5-grams in the input sentence. For each window size, we use N (e.g., 32) filters to learn complementary features from the same word windows. After convolution, *ReLu* activation and *1-max* pooling, the first layer outputs a N-dimensional feature vector for each window size. The three feature vectors output by the first layer are intended to capture the most informative 1-gram, 3-grams and 5-grams in the input sentence, and are used as the input vectors to the second layer. The goal of the second layer is to extract the interactive information of these n-grams, for example, syntactic or semantic dependence of different parts of a sentence. The second layer uses filters with window size $h = 3$. For each feature vector output by the first layer, the second layer uses M (e.g., 32) filters. After convolution, *ReLu* activation and *1-max* pooling, the second layer outputs three M-dimensional feature vectors.

Finally, the output layer (also known as fully connected layer) concatenates the feature vectors output by the second layer, and performs a linear transformation to map the $3M$-dimensional feature vector into a H-dimensional vector in the dual-language vector space. Note that sentences from both languages are represented as language-independent vectors in the dual-language vector space, in which semantically close sentences are likewise close.

3.3.2 Generating Dual-language Sentence Pairs

To train the proposed dual-channel CNN, we need a set of dual-language sentence pairs, each of which is a pair of semantically equivalent or different sentences, one in English and the other in Chinese. In statistical machine translation research, such dual-language documents are usually obtained from human translated documents, for example United Nation documents for different languages [11, 36]. In the software engineering domain, such human-translated dual-language documents rarely exist, except some textbooks

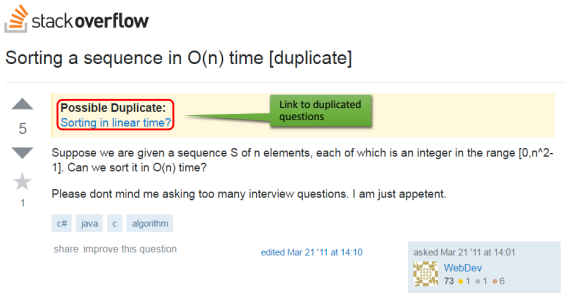


Figure 5: An example of duplicate question

or user manual of software tools. However, for the task of question retrieval, textbook and user manual contents are not sufficient to cover the ways people ask questions and the vocabulary of questions people ask. Furthermore, the key design goal of our approach is to be able to quantify semantic similarity across query-question lingual barrier and lexical gap. To that end, the training sentence pairs must include sufficient examples of semantically equivalent Chinese-English sentence pairs with semantic deviation and lexical gap. Textbook and user manual contents cannot satisfy this need.

We exploit large amounts of Stack Overflow duplicate questions and machine translation to generate dual-language sentence pairs for training our CNN. On Stack Overflow, large amounts of duplicate questions are kept (except exact or nearly exact duplicates), because Stack Overflow acknowledges that users could formulate the same question in different ways[5] (see Figure 5 for an example). These duplicate questions are semantically equivalent as they can be answered by the same answer, but often exhibit lexical gap. We collect question titles of Stack Overflow duplicate questions as English sentences, and translate them into Chinese counterparts. As training of a CNN requires large amounts of dual-language sentence pairs, human translation is impractical. Therefore, we use Google Translate to generate the Chinese translations of Stack Overflow question titles. Although machine translation will likely introduce semantic deviation from the original English questions, there are usually correct translations in a group of duplicate questions. Furthermore, non-English speaking developers are likely to use similar machine translation services, and thus introduce similar translation deviations. Third, certain level of translation deviations in the training sentence pairs will make the CNN more robust to different ways to express semantically similar meanings across languages.

Let EG_i be the ith group of English duplicate questions, and CG_i be the corresponding group of Chinese duplicate questions translated by Google Translate. Denote es_{ij} as the jth English question in the ith English duplicate group, and cs_{ij} as the jth Chinese question in the ith Chinese duplicate group. An English question and its corresponding Chinese translation are indexed by the same ij in the corresponding English and Chinese groups. As illustrated in Figure 6, for a pair of EG_i and CG_i, we generate two sets of semantically equivalent sentence pairs whose semantic similarity is set to 1 (i.e., $sim = 1$). The first set includes the English question title $es_{ij} \in EG_i$ and its corresponding Chinese translation $cs_{ij} \in CG_i$ for all questions in the EG_i and CG_i, such as $es_{11} \in EG_1$ and $cs_{11} \in CG_1$, $es_{12} \in EG_1$ and $cs_{12} \in$

[5]http://stackoverflow.com/help/duplicates

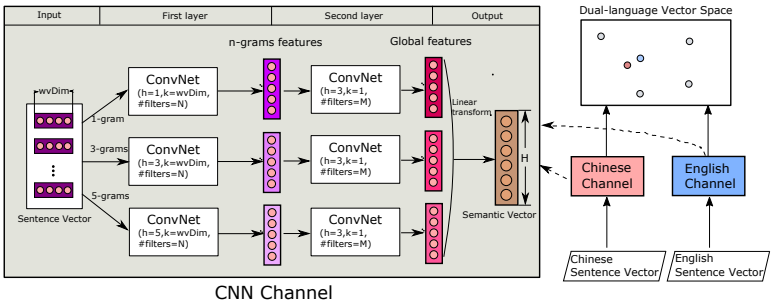


Figure 4: The structure of the proposed CNN channel

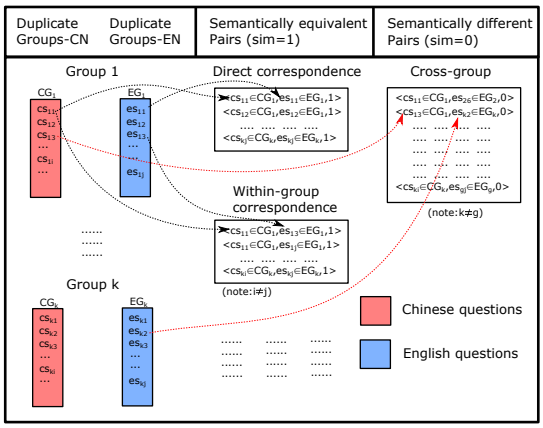


Figure 6: Generating Dual-Language Sentence Pairs

CG_1, etc. The number of entries in this set is the same as the number of duplicate questions in the group. The second set include the English question title $es_{ij} \in EG_i$ and the Chinese translation of a different English question title $cs_{ik} \in CG_i$ $(j \neq k)$, such as $es_{13} \in EG_1$ and $cs_{11} \in CG_1$, $es_{12} \in EG_1$, $cs_{11} \in CG_1$, etc. The entries in the second set are randomly selected within EG_i and CG_i, and the number of the entries in the second set is the same as that of the first set. Both semantically equivalent sets would exhibit semantic deviations caused by machine translation and the second set would exhibit lexical gap between semantically equivalent questions.

We randomly select a set of semantically different sentence pairs, each of which consists of an English question title from one group of English duplication questions EG_k and a Chinese translation from a non-corresponding group CG_g $(k \neq g)$, such as $es_{11} \in EG_1$ and $cs_{26} \in CG_2$. The semantic similarity of these semantically different pairs is set to 0. The number of semantically different sentence pairs across groups is the sum of the two sets of semantically equivalent pairs. These semantically different sentence pairs contrast with the semantically equivalent pairs for training the CNN.

3.3.3 Loss Functions

To train the CNN, we feed the dual-language question pairs to the CNN, giving the English question to English CNN channel and the Chinese question to Chinese CNN channel. Let the kth dual-language question pair be a tuple $\langle es_{k_{eg}k_{eid}} \in EG_{k_{eg}},\ cs_{k_{cg}k_{cid}} \in CG_{k_{cg}},\ sim_k \rangle$, where k_{eg} and k_{cg} are the group index of English duplicate questions and Chinese duplicate questions respectively, and k_{eid} and k_{cid} are the index of the English question and the Chinese question in the respective English- and Chinese-questions group. If k_{eg} and k_{cg} are the same (i.e., $es_{k_{eg}k_{eid}}$ and $cs_{k_{cg}k_{cid}}$ are from the corresponding English-Chinese groups), sim_k is 1, otherwise sim_k=0. The respective CNN channel maps the English or Chinese question as a vector in the dual-language vector space. We denote the two vectors as ev_k and cv_k.

We use *cosine similarity*, i.e., $cos(ev_k, cv_k)$, to measure the distance between the two vectors. Correspondingly, *mean square error* is used as the loss function to quantify the agreement between the computed similarity and the expected similarity:

$$L_{cos} = \frac{1}{N_{ds}} \sum_{1 \leq k \leq N_{ds}} (sim_k - cos(ev_k, cv_k))^2$$

where N_{ds} is the number of dual-language question pairs for training. Mean square error of cosine similarity (i.e., cosine loss) will guide the CNN to map semantically equivalent questions around the same angle in the vector space (thus cosine similarity would be close to 1), but map semantically different questions at 90 degrees (thus cosine similarity would be close to 0).

Meanwhile, a SVM is introduced for the two output vectors ev_k and cv_k respectively. In our case, there are hundreds of thousands of duplicate-questions groups. Such a large-class classification problem will make the SVM training time-consuming. Since the SVM here is used as an auxiliary driving force to guide the CNN to capture appropriate sentence features for separating semantically different groups, performing a sampled classification is sufficient, similar to the negative sampling method used in training word embeddings [20]. The input of the sampling SVM is the output vector from the CNN, ev_k (or cv_k). A linear layer is used to calculate the scores of the vector ev_k (or cv_k) in different duplicate-questions group j:

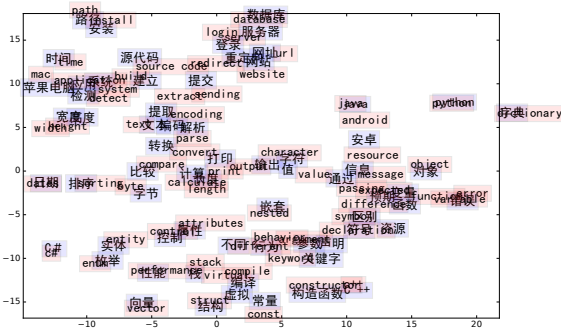
$$s_j = u_j^T ev_k + b_j$$

where u_j and b_j are the weight and bias parameter of the SVM. Then, the sampling multiclass SVM loss is defined as:

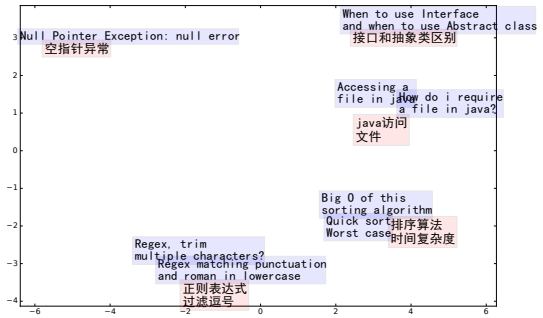
$$L_{SVM}(ev_k) = \sum_{j \in corrupt(g)} max(0, s_j - s_{k_{eg}} + 1)$$

where the $corrupt(g)$ is a set of indexes of a small number (e.g. 10) of sampled corrupted duplicate-questions groups. A corrupted group means the group other than the ground-truth group for the vector ev_k (or cv_k), i.e., k_{eg} (or k_{cg}). The $L_{svm}(cv_k)$ can be defined in the same way. The overall loss function for the sampling SVM is:

$$L_{SVM} = \frac{1}{N_{ds}} \sum_{1 \leq k \leq N_{ds}} (L_{SVM}(ev_k) + L_{SVM}(cv_k))$$



(a) 2D-visualization of semantically similar words



(b) 2D-visualization of semantically similar sentences

Figure 7: Words or sentences in the dual-language space

As we only sample a small number of corrupted groups for computing SVM loss, the computation time is dramatically reduced. Although the sampling SVM provides only an approximation to the full SVM, our experiments show that when it cooperates with cosine loss function, the sampling SVM guides the CNN to cluster semantically equivalent sentences within small Euclidean distance, but separates semantically different sentences by large Euclidean distance.

3.3.4 Training Process

Intuitively, the training process makes the CNN learn to capture the most informative features of English and Chinese sentences for quantifying semantic similarity across lingual barrier and lexical gap. Formally, the training objective is to minimize the loss function overall the entire set D of dual-language sentence pairs:

$$\arg\min_{W,b} \; L_{cos} + L_{SVM} + R(W)$$

where $R(W)$ is *l2 norm* constraint, which is used to avoid overfitting problem. The parameters (W, b) to be estimated include: the filters and bias terms in the first and second layer of the CNN, the linear transformation matrix of the output layer, and the parameters of the SVM classifiers. The Adam update algorithm [15] is used instead of stochastic gradient decent (SGD) for a faster convergence.

3.4 Cross-Lingual Question Retrieval

After training the CNN, we learn the parameters of each layer of the CNN. It can then be used to map questions and queries in either language onto a dual-language vector space. Figure 7b shows the visualization of several semantically similar Chinese queries and English questions that uses the t-SNE dimensional reduction technique [37]. Figure 7a gives some most frequently used English words in Stack Overflow posts and their corresponding Chinese translations in the two-dimensional t-SNE visualization [37]. We can see that semantically close terms and sentences from both languages are close in the dual-language vector space. This shows the potential usefulness of our CNN for cross-lingual information retrieval.

For Chinese-to-English question retrieval, we can collect a knowledge base of English questions (different from those used to train the CNN). These English questions will be converted into word vector representations using English word embeddings, and then question vectors will be mapped onto the dual-language vector space using the English CNN channel. Users can query these English questions using queries in Chinese. Chinese queries do not need to be translated. Instead, they are converted to word vector representations using Chinese word embeddings, and then query vectors will be mapped onto the same dual-language vector space using the Chinese CNN channel. Most relevant English questions can be retrieved by finding the k-nearest questions close to the query in the vector space.

Although this work focuses on Chinese-to-English question retrieval, our CNN can also support English-to-English, Chinese-to-Chinese, English-to-Chinese retrieval. That is, queries in either language can retrieve questions in either language. Depending on the language of queries and questions, the respective CNN channel can be used to map queries and questions onto the same vector space, then question retrieval can be done by finding k-nearest neighbors.

4. EVALUATION

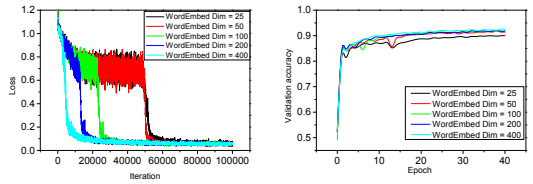
We perform a set of experiments to answer the following questions:

- How do the hyperparameters of the CNN affect the performance of the CNN?
- Can our approach outperform the query-translation-then-monolingual-question-retrieval method and in what cases?
- Can the CNN trained with Stack Overflow data be used to quantify semantic similarity of English-Chinese sentences from a different data source?

4.1 Dataset

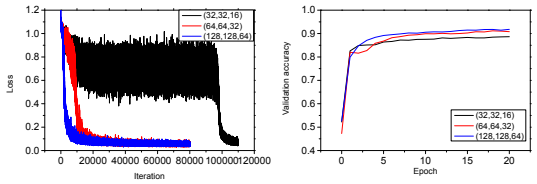
The main data source of our study is Stack Overflow data dump of January 2016[6]. We collect 0.3 million duplicate questions from the data dump, which form 0.1 million duplicate-questions groups. We also crawl 0.1 million Chinese questions from two Chinese Q&A sites (SegmentFault and V2EX). We augment the Chinese questions with the Chinese translations of 0.2 million randomly selected Stack Overflow questions. In this work, we use only question titles, each of which is treated as a sentence. The corpus of English sentences and Chinese sentences is used to learn monolingual word embeddings to represent English and Chinese words, respectively. We randomly divide the duplicate-questions groups into three subsets, i.e., 80% (training), 10% (validation) and 10% (test). The 80% subset is used to generate

[6]https://archive.org/download/stackexchange



(a) Loss convergence (b) Validation accuracy

Figure 8: Impact of word-embedding dimensionality



(a) Loss convergence (b) Validation accuracy

Figure 9: Impact of the CNN complexity

dual-language sentence pairs for training the CNN. The first 10% subset is used to tune the CNN hyperparameters during training. The second 10% subset is used to compare our approach with the baseline method.

4.2 CNN Hyperparameter Optimization

Our CNN has two hyperparameters: the dimensionalities of the word embeddings and the output feature vectors of each layer. We conduct experiments to investigate the influence of different parameter choices.

4.2.1 The Dimensionality of Word Embeddings

In this experiment, we set the dimensionality of word embeddings at 25, 50, 100, 200 and 400. We use the setting (64, 64, 32) for the dimensionality of output feature vectors of the three layers in our CNN (see Section 4.2.2).

Figure 8a plots the convergence of the loss function by the number of CNN training iterations for different dimensionalities. The loss function converges at different speed for different dimensionalities. The larger the dimensionality is, the fewer iterations the loss function takes to converge. However, this does not mean that loss function takes less time to converge for larger dimensionality, because for a larger dimensionality it will need more computation time per iteration.

Figure 8b plots the validation accuracy by the number of epochs for different dimensionalities. One epoch consists of one full training cycle on the training set. Given a pair of English-Chinese sentences in the validation data, if the cosine similarity of the two sentences in the dual-language vector space is above 0.5, we consider the two sentences as semantically equivalent, otherwise as semantically different. Validation accuracy is computed by the number of accurate predictions of semantically equivalent (or different) sentence pairs in the validation set. As the validation dataset has an equal number of semantically equivalent (or different) sentence pairs, the accuracy of a random prediction would be around 0.5. Figure 8b shows that the validation accuracy increases and converges at very similar rate for different dimensionalities, and word-embedding dimensionality does not have big impact on the accuracy of semantic-equivalence/difference prediction.

4.2.2 The Dimensionality of Feature Vectors

The dimensionality of output feature vectors of CNN layers affects the complexity of the CNN. In our CNN, the dimensionality of feature vectors in the first layer is proportional to the number of filters (N) used for each window size. The dimensionality of feature vectors in the second layer is proportional to the number of filters (M) used for each input vector. The dimensionality of feature vector in the output layer is H. The search space of these CNN parameters is extremely large. We experiment with three settings (N, M, H): (32,32,16), (64,64,32), (128,128,64), with increasing complexity. In this experiment, we use 200-dimensional word embeddings. Figure 9a and Figure 9b plot the loss function convergence and validation accuracy for different CNN complexities. We can see that the more complex the CNN is, the fewer iterations the loss function takes to converge, and the slightly higher the validation accuracy is.

4.3 Comparison of CLQR Performance

In this section, the comparison experiments with the baseline methods are shown and discussed.

4.3.1 Experiment Setup

Ground truth: For the 10% duplicate-question groups used as test data, we compile all English questions as the knowledge base. Let EG_i be a group of English duplicate questions in the test data, and CG_i be the corresponding group of Chinese duplicate questions (see Section 3.3.2). We use each Chinese question $cs_{ij} \in CG_i$ as a query to retrieve English questions $es_{ik} \in EG_i$ $(k \neq j)$. That is, all English questions in EG_i, except the one corresponding to the query Chinese question cs_{ij}, are used as ground truth to evaluate the performance of question retrieval.

Baseline methods: For the baseline method, we use Google Translate to translate the Chinese query into an English query, and then use Lucene to retrieve English questions for the translated English query. In addition to Google-Translate+Lucene, we also directly use the English question corresponding to the query Chinese question as the English query for Lucene search. This second method can be considered as the "perfect" translation of the Chinese query for monolingual question retrieval.

Our approach: Based on the CNN hyperparameters experiments, we use *200-dimensional word embeddings* and the output feature vectors setting *(128,128,64)* in this comparative study. To gain insight into the impact of our loss functions on the CNN performance, we train two CNNs, one with only cosine loss (referred to as CNN (only cos)) and the other with both cosine loss and SVM loss (referred to as CNN (cos+SVM)).

Metrics: Given a Chinese query, let RE be the list of retrieved questions for the query, and let GT be the set of ground truth questions for the query. We evaluate the question-retrieval performance of our approach and the baseline methods using three metrics: Precision@k (Pr@k), Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR). Precision@k refers to the fraction of retrieved questions that are relevant in the top-k retrieval results, i.e., $|RE \bigcap GT|/k$. Let the rank position of all relevant questions in the RE for a query be r_1, r_2, ..., r_z. We can compute Pr@k for $k = r_1, r_2, \ldots, r_z$. The average precision for this query is defined as the mean of the computed Pr@k. Mean Average Precision (MAP) is the mean of the average precisions over

all the queries. Let k be the rank position of the first relevant question in the RE for a query, then the reciprocal rank (RR) is defined as $\frac{1}{K}$. Mean Reciprocal Rank (MRR) is the mean of the RRs over all queries.

4.3.2 Quantitative Results

Table 1 shows the performance results. We can see that:

- The performance of GoogleTranslate+Lucene is worse than that of Lucene-only (i.e., simulating "perfect" translation). This suggests that semantic deviations in query translation can degrade the performance of subsequent monolingual retrieval step.
- In contrast, both our CNN settings outperform GoogleTranslate+Lucene baseline, and our CNNs achieve almost the same or better performance than Lucene-only baseline. This suggests that directly quantifying semantic similarity between Chinese queries and English questions in the dual-language vector space is more robust than the separate query translation followed by monolingual retrieval.
- CNN (cos+SVM) setting significantly outperform CNN (only cos) setting on all performance metrics. This suggests that considering group information of semantically equivalent or different sentence pairs can help CNN capture richer semantic features of corresponding English and Chinese sentences, which in turn can improve the performance of question retrieval.
- The Pr@1 and MRR of our CNN (only cos) setting (the poorer performer in the two CNN settings) are almost the same as those of Lucene-only baseline. This suggests that our CNN (only cos) can achieve comparable performance as the traditional IR method in terms of the rank of the first relevant question. However, the Pr@5, Pr@10 and MAP of our CNNs (both settings) are significantly better than the corresponding metrics of the baseline methods. Our analysis of the retrieval results suggests that our CNNs can retrieve more relevant questions and rank them higher, especially those with lexical gaps to the query, which traditional IR methods often fail to retrieve, as discussed below.

4.3.3 Qualitative Analysis of Retrieval Results

To gain insight into the capability of our CNN and compare results from two methods more objectively, we randomly select 30 queries for which the Lucene only baseline has a better MAP than our CNN (cos+SVM), and another 30 queries for which our CNN (cos+SVM) has a better MAP. We manually compare the question retrieval results by our CNN (cos+SVM) with the Lucene only baseline for each selected query. Table 2 presents one query in which Lucene-only baseline returns the better results, and two queries in which CNN (cos+SVM) is better. Note that all queries are in Chinese and the English translation is for non-Chinese readers as references.

Because a group of duplicate questions often has lexically very similar question titles, Lucene can successfully retrieve other questions given one of the questions in the group and also rank these question highly in the results list (as for the first query in Table 2). In such cases, our approach can achieve the overall comparable performance as the Lucene method. However, for questions that have lexical gap to the queries, it is often hard for Lucene to determine the query-question similarity. As a result, the performance of Lucene degrades because such questions cannot be ranked highly due to few words in common with the query. In contrast, our CNN can capture semantic similarity in the face of lexical gap between queries and questions. Therefore, it often outperforms the Lucene baseline for duplicate-question groups with several questions with lexical gaps.

Take the second query in Table 2 as an example, the top-3 results by the Lucene-only baseline cover most words in the query. However, the semantics of these questions are totally different to the query. In contrast, our CNN can capture the semantic similarity between the query and the questions in the knowledge base, hence even if the relevant questions are lexically very different from the query, our CNN can still successfully retrieve them and rank them high in the list, such as the first two questions retrieved by our CNN for the second query. Similarly, the Lucene baseline retrieves three lexically very similar but semantically different questions for the third query. Note that it fails to capture the semantics of the words "Python" and "MySQL". In contrast, the top-3 questions our CNN returns are all relevant to "Python" and "MySQL", even though these questions are lexically very different from the query.

Although this analysis is anecdotal, our empirical observations reflect the design intuition of our deep learning system, and echo the application results of word embeddings and CNN in other IR and NLP tasks [6, 10, 14].

4.4 Domain Adaptation of Our CNN

Dataset: We crawl the official Python tutorial at http://python.usyiyi.cn. This website provides both the English version of the tutorial and the Chinese version. Each English sentence has a corresponding Chinese sentence translated by human. Given this bilingual parallel corpus, we consider a pair of corresponding English-Chinese sentences as the semantically equivalent pairs, which means $sim = 1$ in our setting. Besides, we randomly select the same number of English-Chinese sentence pairs that do not have correspondences as semantically different sentence pairs, which means that $sim = 0$ for these pairs.

Results: Given a pair of Chinese and English sentences collected from the website, we use the English and Chinese CNN channel trained using the Stack Overflow text to map the sentences onto a dual-language vector space. If the cosine similarity of the two sentences in the vector space is above 0.5, we consider the two sentences as semantically equivalent, otherwise, as semantically different. We compare the prediction results by our CNN with the ground truth label of the two sentences. The accuracy of the correct prediction by our CNN is 83%, which is acceptable but moderately lower than the prediction accuracy for the Stack Overflow test data (which is 92%). This experiment suggests that Stack Overflow text has a good coverage of software engineering knowledge such that word embeddings and CNN trained using Stack Overflow text can be extended to recognize and quantify semantics of software engineering text from a very different data source. However, the performance may degrade due to certain unseen terms and/or semantics across different data source. For optimal performance, our approach can be extended to incorporate data from different sources into word embeddings learning and CNN training.

Table 1: Comparison of cross-lingual question-retrieval performance by different methods

Type of Method	Query Language	Pr@1	Pr@5	Pr@10	MAP	MRR
Lucene only	English	0.429	0.361	0.319	0.103	0.549
GoogleTranslate+Lucene	Chinese	0.386	0.313	0.273	0.095	0.503
CNN (only cos)	Chinese	0.430	0.419	0.407	0.266	0.549
CNN (cos+SVM)	Chinese	**0.504**	**0.483**	**0.468**	**0.289**	**0.617**

Table 2: Retrieval using GoogleTranslate (GT)+Lucene and CNN (cos+SVM) (Italic font for relevant results)

Query	对ArrayList的对象属性排序 (Sort ArrayList of custom Objects by property?)	用Python在终端打印出有颜色的文字? (Print in terminal with colors using Python?)	在Python中如何连接mysql数据库 (How do I connect to a MySQL Database in Python?)
GT+ Lucene	1.*Sorting ArrayList of Objects by Object attribute* *2.ArrayList sorting on the basis of object property* 3.*Sorting an object ArrayList by an attribute value in Java*	1.color text in terminal aplications in unix 2.Print text from the terminal on a label? 3.C++ change terminal text output color	1.How do I connect to an Oracle Database in R? 2.How do I connect to an SQLite database with PHP? 3.How to connect to a localhost MySQL database
CNN (cos+SVM)	1.*Sorting ArrayList of Objects by Object attribute* 2.*ArrayList object sorting confusion* 3.Sort ArrayList by its properties	1.*change color of individual print line in Python3.2?* 2.*is it possible to add colors to python output?* 3.Printing an upside down triangle in Python 3.4.2?	1.*Python 3.4.0 with MySQL database* 2.*Python logging to mysql* 3.*Is it possible to hook up a MySQL database with python*

5. RELATED WORK

In this section, we introduce studies of information retrieval (IR) especially cross-lingual retrieval in Software Engineering. Then, we describe the application of deep learning in Software Engineering.

5.1 IR in Software Engineering

Information retrieval is an important task in Software Engineering and it has been mainly studied from two aspects, code search and documentation search. Many code search engines have been proposed based on program patterns [23], test cases [18], program semantics [27], and user feedback [39]. For software documentation, Siddharth et al. [35] build a plugin to find API documentation for APIs mentioned in Q&A discussions. Robillard and Chhetri [29] can recommend fragments of API documentation potentially important to a programmer using an API. Information retrieval has also been used in bug localization [31], traceability recovery [4], and feature location [9, 38].

With the advent of Web 2.0, Q&A websites have become an important information source for developers. Hence, much work has been carried out on Q&A data, such as ranking answers based on user feedback [8] and automatically answering interrogative questions [47] (e.g.,how to and why questions). Compared with these works, we focus on bilingual information retrieval in Q&A site which is more challenging than monolingual analysis.

Some work has been done on cross-lingual issues in Software Engineering, such as bug localization between Chinese and English [41], traceability recovery between Italian and English [12]. Xu et al. [42] propose a domain-specific bilingual question retrieval system which customizes the general translation by finding software-specific terms in a domain specific corpus to enhance the translation performance. These existing methods is rule-based, relying on human-crafted features and general machine translation system.

In contrast, given a Chinese query, our goal is to retrieve relevant English questions in which no query or question translation is required. We adopt deep learning to automatically learn latent word and sentence features, without the need for human-engineered heuristics or rules. Inspired by the recent work by Bogdanova et al [3], we use CNNs to learn to quantify semantic similarity across lingual barrier and lexical gap from large amounts of duplicated questions and their corresponding Chinese translations. As a result, our approach for quantifying semantic similarity of bilingual documents is more robust, scalable and extensible than traditional translation-based IR systems, even in the face of semantic deviations and lexical gap between documents.

5.2 Deep Learning in Software Engineering

Recently, deep learning has made a breakthrough in many areas such as computer vision [16], speech recognition [13] and natural language processing [33, 43, 45]. In software engineering, some work also employs deep learning in software-specific tasks. Chen et al [5] incorporate word embeddings and categorical and relational knowledge to find analogical libraries across different programming languages. White et al [40] use recurrent neural networks to achieve a higher accuracy in code sequence auto-completion than the traditional n-gram model. Mou et al [21, 22] propose a tree-based CNN to learn the structure of the program which is useful for program analysis such as classifying programs and detecting code patterns. Reed and Freitas [25] adopt a recurrent and compositional neural network to learn to predict outputs of simple programs without executing the programs.

Our approach is motivated by the success of the deep learning methods in software engineering and natural language processing. In this work, we design deep learning techniques to tackle a challenging research problem, i.e, domain specific cross-lingual question retrieval, which could narrow the gap between millions of non-English speaking developers and the tremendous English content in Stack Overflow.

6. CONCLUSION AND FUTURE WORK

In the paper, several techniques including word embeddings, CNN, sampling SVM are innovatively incorporated together to build a deep learning system that bridges the language gap by directly quantify query-questions semantic similarity across languages. The system transfers the cross-lingual question retrieval problem into a k-nearest neighbors search problem in a dual-language vector space in which no query or question translation is required. Our experiments show that our system is especially effective to recognize and quantify query-questions semantic similarity across semantic deviations and lexical gaps, which is the major barrier for traditional translation-based and word-overlapping based retrieval method. This is because the underlying word embeddings and CNN models can better represent query and questions at semantic level (even across lingual barrier), rather than relying on the lexical similarity of word overlaps.

We are implementing our system as an online service which could benefit millions of Chinese developers who want to reuse English knowledge in Stack Overflow. The design of our system is general and can be extended to any other languages. It would be interesting to see whether our approach can effectively support multi-lingual question retrieval.

Acknowledgments. This work was partially supported by Singapore MOE AcRF Tier-1 grant M4011448.020.

7. REFERENCES

[1] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

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

[3] D. Bogdanova, C. dos Santos, L. Barbosa, and B. Zadrozny. Detecting semantically equivalent questions in online user forums. *CoNLL 2015*, page 123, 2015.

[4] G. Capobianco, A. D. Lucia, R. Oliveto, A. Panichella, and S. Panichella. Improving ir-based traceability recovery via noun-based indexing of software artifacts. *Journal of Software: Evolution and Process*, 25(7):743–762, 2013.

[5] C. Chen, S. Gao, and Z. Xing. Mining analogical libraries in q&a discussions -incorporating relational and categorical knowledge into word embedding. In *23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 338–348. IEEE, 2016.

[6] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[7] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[8] D. H. Dalip, M. A. Gonçalves, M. Cristo, and P. Calado. Exploiting user feedback to learn to rank answers in q&a forums: a case study with stack overflow. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 543–552. ACM, 2013.

[9] B. Dit, M. Revelle, and D. Poshyvanyk. Integrating information retrieval, execution and link analysis algorithms to improve feature location in software. *Empirical Software Engineering*, 18(2):277–309, 2013.

[10] C. N. dos Santos and M. Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78, 2014.

[11] A. Eisele and Y. Chen. Multiun: A multilingual corpus from united nation documents. In *LREC*, 2010.

[12] J. H. Hayes, H. Sultanov, W.-K. Kong, and W. Li. Software verification and validation research laboratory (svvrl) of the university of kentucky: traceability challenge 2011: language translation. In *Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering*, pages 50–53. ACM, 2011.

[13] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

[14] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

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

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[18] O. A. L. Lemos, S. K. Bajracharya, J. Ossher, R. S. Morla, P. C. Masiero, P. Baldi, and C. V. Lopes. Codegenie: using test-cases to search and reuse source code. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pages 525–526. ACM, 2007.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[20] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[21] L. Mou, G. Li, Y. Liu, H. Peng, Z. Jin, Y. Xu, and L. Zhang. Building program vector representations for deep learning. *arXiv preprint arXiv:1409.3358*, 2014.

[22] L. Mou, G. Li, L. Zhang, T. Wang, and Z. Jin. Convolutional neural networks over tree structures for programming language processing. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[23] S. Paul and A. Prakash. A framework for source code search using program patterns. *Software Engineering, IEEE Transactions on*, 20(6):463–475, 1994.

[24] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM, 1998.

[25] S. Reed and N. de Freitas. Neural programmer-interpreters. *arXiv preprint arXiv:1511.06279*, 2015.

[26] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[27] S. P. Reiss. Semantics-based code search. In *Proceedings of the 31st International Conference on Software Engineering*, pages 243–253. IEEE Computer Society, 2009.

[28] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al. Okapi at trec-3. *NIST SPECIAL PUBLICATION SP*, 109:109, 1995.

[29] M. P. Robillard and Y. B. Chhetri. Recommending reference api documentation. *Empirical Software Engineering*, 20(6):1558–1586, 2015.

[30] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[31] R. K. Saha, M. Lease, S. Khurshid, and D. E. Perry.

Improving bug localization using structured information retrieval. In *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, pages 345–355. IEEE, 2013.

[32] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[33] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[34] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[35] S. Subramanian, L. Inozemtseva, and R. Holmes. Live api documentation. In *Proceedings of the 36th International Conference on Software Engineering*, pages 643–652. ACM, 2014.

[36] J. Uszkoreit, J. M. Ponte, A. C. Popat, and M. Dubiner. Large scale parallel document mining for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1101–1109. Association for Computational Linguistics, 2010.

[37] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[38] J. Wang, X. Peng, Z. Xing, and W. Zhao. Improving feature location practice with multi-faceted interactive exploration. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 762–771. IEEE Press, 2013.

[39] S. Wang, D. Lo, and L. Jiang. Active code search: incorporating user feedback to improve code search relevance. In *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, pages 677–682. ACM, 2014.

[40] M. White, C. Vendome, M. Linares-Vásquez, and D. Poshyvanyk. Toward deep learning software repositories. In *Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on*, pages 334–345. IEEE, 2015.

[41] X. Xia, D. Lo, X. Wang, C. Zhang, and X. Wang. Cross-language bug localization. In *Proceedings of the 22nd International Conference on Program Comprehension*, pages 275–278. ACM, 2014.

[42] B. Xu, Z. Xing, X. Xia, D. Lo, Q. Wang, and S. Li. Domain-specific cross-language relevant question retrieval. In *Proceedings of the 13th International Conference on Mining Software Repositories*, page (To appear). IEEE, 2016.

[43] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

[44] Y. Zhang and B. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820, 2015.

[45] X. Zheng, H. Chen, and T. Xu. Deep learning for chinese word segmentation and pos tagging. In *EMNLP*, pages 647–657, 2013.

[46] G. Zhou, T. He, J. Zhao, and P. Hu. Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of ACL*, pages 250–259, 2015.

[47] Y. Zou, T. Ye, Y. Lu, J. Mylopoulos, and L. Zhang. Learning to rank for question-oriented software text retrieval (t). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 1–11. IEEE, 2015.