

Domain-specific cross-language relevant question retrieval

Bowen Xu¹ · Zhenchang Xing² · Xin Xia^{3,4}  · David Lo⁵ · Shanping Li¹

© Springer Science+Business Media, LLC 2017

Abstract Chinese developers often cannot effectively search questions in English, because they may have difficulties in translating technical words from Chinese to English and formulating proper English queries. For the purpose of helping Chinese developers take advantage of the rich knowledge base of Stack Overflow and simplify the question retrieval process, we propose an automated cross-language relevant question retrieval (*CLRQR*) system to retrieve relevant English questions for a given Chinese question. *CLRQR* first extracts essential information (both Chinese and English) from the title and description of the input Chinese question, then performs domain-specific translation of the essential

Communicated by: Romain Robbes, Christian Bird and Emily Hill

✉ Xin Xia
xxia@zju.edu.cn; Xin.Xia@monash.edu

Bowen Xu
max_xbw@zju.edu.cn

Zhenchang Xing
Zhenchang.Xing@anu.edu.au

David Lo
davidlo@smu.edu.sg

Shanping Li
shan@zju.edu.cn

¹ College of Computer Science and Technology, Zhejiang University, Hangzhou, China

² Research School of Computer Science, Australian National University, Canberra, Australia

³ Faculty of Information Technology, Monash University, Melbourne, Australia

⁴ Australia College of Computer Science and Technology, Zhejiang University, Hangzhou, China

⁵ School of Information Systems, Singapore Management University, Singapore, Singapore

Chinese information into English, and finally formulates an English query for retrieving relevant questions in a repository of English questions from Stack Overflow. We propose three different retrieval algorithms (word-embedding, word-matching, and vector-space-model based methods) that exploit different document representations and similarity metrics for question retrieval. To evaluate the performance of our approach and investigate the effectiveness of different retrieval algorithms, we propose four baseline approaches based on the combination of different sources of query words, query formulation mechanisms and search engines. We randomly select 80 Java, 20 Python and 20 .NET questions in SegmentFault and V2EX (two Chinese Q&A websites for computer programming) as the query Chinese questions. We conduct a user study to evaluate the relevance of the retrieved English questions using *CLRQR* with different retrieval algorithms and the four baseline approaches. The experiment results show that *CLRQR* with word-embedding based retrieval achieves the best performance.

Keywords Domain-specific translation · Cross-language question retrieval

1 Introduction

Domain-specific community Q&A websites, such as Stack Overflow, have become a prevalent platform for knowledge sharing and acquisition. In the past 7 years, Stack Overflow has accumulated over 10 million questions, and has become a tremendous knowledge repository of developers' thoughts and practices. According to Stack Overflow 2016 Developer Survey,¹ about 40 million people visit Stack Overflow monthly. Every 8 seconds or so, a developer asks a question on Stack Overflow and 56,033 coders in 173 countries answer the call. To allow developers in non-English speaking countries to participate on Stack Overflow, Stack Overflow has launched localized versions of Stack Overflow in Portuguese, Russian, and Japanese.

Ten percent of the world's programmers are in China.² Even without a localized version of Stack Overflow in Chinese, it would still be very desirable to support developers in China to easily access the knowledge repository of the English version of Stack Overflow. Developers in China usually graduate with a Bachelor degree. To fulfill the degree requirements, they need to pass the national college English test (Level 4) which contains a very short writing task but mainly covers listening, reading and general translation tasks. As such, developers in China are often equipped with basic English reading comprehension skills to read posts in English. However, most of them often are not comfortable to ask questions in English (Harkness 2017; Jui 2010). This makes it difficult for them to formulate English queries to search the Internet.

This reality of English reading and writing skills of developers in China indicates a potential to make the content of the English version of Stack Overflow more easily accessible to developers in China. In this paper, we propose a domain-specific cross-language relevant question retrieval (*CLRQR*) system that takes as input a question written in Chinese and

¹Stack Overflow 2016 Developer Survey, <https://stackoverflow.com/research/developer-survey-2016>

²Planet Earth Has 18.5 Million Developers, <http://www.drdoobs.com/tools/planet-earth-has-185-million-developers/240165016>

returns relevant questions written in English from Stack Overflow. These relevant English questions are the keys to accessing the knowledge in the English version of Stack Overflow.

Using our approach, developers can write questions in Chinese (may be mixed with English words such as programming languages, tools, parameters). Given a question, our *CLRQR* system first extracts essential information (both Chinese and English) from the title and description of the input Chinese question, then performs domain-specific translation of the essential Chinese information into English, and finally formulates an English query with scored English words. We propose a suitable retrieval algorithm for retrieving relevant English questions from Stack Overflow. Our *CLRQR* system improves the efficiency of Chinese developers to find solutions in a repository of English questions by solving the problem of domain-specific translation of technical words and the problem of query formulation. A key benefit of our approach is that it allows Chinese developers to more easily take advantage of high-quality English Q&A resources on Stack Overflow.

The main contributions of this paper are the following (the new contributions that extend our previous works (Xu et al. 2016, 2017b) are highlighted in bold font):

1. We introduce a cross-language question retrieval approach to retrieve relevant English questions for a Chinese question.
2. Based on the term frequency of Stack Overflow questions for a particular domain (e.g., Java, Python, or .NET), we build a domain-specific vocabulary to optimize the translation results of general Chinese-to-English translation tool for the domain. Based on this domain-specific vocabulary, we propose a filter method to select more-likely domain-specific translations.
3. We combine the two keyword extraction algorithms to improve the accuracy of Chinese keyword extraction from the description of query Chinese questions.
4. In addition to the word-matching based question retrieval algorithm proposed in our previous work (Xu et al. 2016), **we design two more question retrieval algorithms based on different word representations and similarity metrics, namely word embeddings and vector space model. We extend our experiment to investigate the effectiveness of different retrieval algorithms.**
5. **We evaluate our proposed domain specific translation method by comparing it with the translation method which uses the most common translation result without considering the domain-specific meaning of the translations.**
6. We conduct an evaluation of our approach and the four baseline approaches on 80 Java questions in Chinese from *SegmentFault* and *V2EX*. **To demonstrate the generality of our approach, we also conduct experiments on 20 Python questions and 20 .NET questions in Chinese from *SegmentFault* and *V2EX*.**
7. **We investigate the impact of different weight settings for different kinds of essential information on the performance of question retrieval.**
8. We conduct qualitative analysis of the experiment results and **report a set of examples to illustrate the benefits of our approach.**

The remainder of this paper is structured as follows. Section 2 presents a motivating example and elaborates the challenges for cross-language relevant question retrieval. Section 3 describes the overall framework and the details of our proposed approach. Section 4 introduces our experimental methods. Section 5 presents our experiments result. Section 6 discusses the qualitative analysis of some search results, and threats to validity of our proposed approach. Section 7 reviews related work. Section 8 concludes our work and discusses our future plan.

segmentfault 输入关键字搜索 问答 文章 笔记 职位 ...

问答 > 问答详情

代码审查工具

诚实可爱小胖君 304 6月29日 提问

▲ 0 ▼

如题，有没有好一点的开源的java代码审查工具。
适用于java 和javaweb项目。

java 代码审查 工具 开源软件

链接 评论 更多 ▾

Fig. 1 A Chinese Question on *SegmentFault*

2 Motivating Example and Design Challenges

In this section, we present a motivating example to illustrate how our approach formulates an English query from an input Chinese Java question and how it retrieves relevant English Java questions from Stack Overflow. Using this example, we highlight the challenges in cross-language question retrieval and summarize our solutions.

2.1 Motivating Example

Figure 1 shows a Chinese Java question (i.e., tagged with *java*) from SegmentFault (a Chinese Q&A website for computer programming). This Chinese question asks for some useful code review tools which can be used for java and javaweb projects. Our *CLRQR* system first extracts essential Chinese and English information from the question, e.g., the Chinese words “项目”, “开源”, “代码”, “审查”, “工具” and the English words “java”, “javaweb”. Because all the questions in this work are Java-related questions, we consider the word “java” not an important word to distinguish the core issues of different questions. Therefore, we discard the English word “java” and keep only the English word “javaweb”. Then the *CLRQR* system performs domain-specific translation of the extracted Chinese words, and formulates a English query (i.e. a set of 6 English words with scores as shown in Table 1) that an English-speaking developer may use for the similar questions.

Table 1 Word List with Score

Word	Score
code	1.20
review	1.20
tool	1.20
javaweb	1.00
project	0.20
opensource	0.20

Our user study (detailed in Section 4.4) shows that users consider this query accurate and useful in helping them retrieve relevant English questions for the given Chinese question. Given the English query, our *CLRQR* system can use three different question retrieval algorithms to calculate the relevance between the English query and a repository of English Java questions from Stack Overflow. All retrieval algorithms return the top-10 most relevant English Java questions for the given Chinese question. Figure 2 shows one of the top-10 most relevant English Java questions which can be retrieved by all the algorithms for the given Chinese Java question in Fig. 1. This English question also asks for some useful code review tools which can be used for java. Through the user evaluation, we find that the retrieved English questions are relevant for the Chinese question and it is useful in helping developers solve the problem in the Chinese question.

2.2 Design Challenges

Cross-language relevant question retrieval is a very complex process. To achieve the above objective of the cross-language question retrieval, we must address the following three challenges:

2.2.1 Challenges in Keyword Extraction

A question may contain a lot of words. We would like to extract the essential information for query formulation. To that end, we should use keyword extraction algorithms to summarize the essential information in the question. Many keyword extraction algorithms have been proposed in the natural language processing field. Different algorithms are based on different heuristics to evaluate the importance of a word. As they are heuristic-based, some keyword extraction algorithms may perform better than others on some cases, but worse on



stackoverflow Questions Tags Users

What is the best Java Code Review tool to add as plugin to eclipse?

Add  projects to your  **stackoverflow** profile. CAREERS

▲ I want to know what is the best Java code review tool available. Which tool is most preferred and which all can be added as eclipse plug-in.

2

▼ I tried PMD, but somehow I did not find it that good. I also tried to install Find-Bug, but having some issues with the eclipse plug-in installation..may be it's official site is not maintained any more..

★

2 What are the other good tools available, I check the [Best free Code Review tool for Eclipse/Java/Flex Development \[closed\]](#), but it's a quite old post, so not sure if the content still holds good after 3 years..

java eclipse plugins code-review

Fig. 2 A Relevant English Question on *Stack Overflow*

other cases. One way to address the weaknesses of these keyword extraction algorithms is to combine them together in order to make a comprehensive judgment. In our *CLRQR* system, we use two different keyword extraction algorithms (FudanNLP³ and IctclasNLP⁴) to extract Chinese keywords in the title and description of the Chinese question, and take the union of the two sets of keywords as the final Chinese keywords.

2.2.2 Challenges in Domain-Specific Translation

Cross-language question retrieval has to translate the words in the source language into some appropriate words in the target language. The accuracy of this translation will directly affect the relevance of the questions retrieved in the target language. Kluck and Gey (2001b) point out that in many cases there exists a clear difference between the domain-specific meaning and the common meaning of a word. This means that it can be difficult to use the common translation result of a word for domain-specific information retrieval. For example, for the Chinese word “代码”, the general domain translation tool returns several English translations, such as “code” and “word”. In the context of software engineering, the translation “code” is more appropriate than the other translations. Similarly, for the Chinese word “审查”, the translation “review” is more appropriate than the translations “investigate” or “examine”.

Several studies propose domain-specific translation techniques which are based on building domain-specific dictionary (Resnik and Melamed 1997; Hiemstra et al. 1997; Peñas et al. 2012). A few research studies have been carried out on domain-specific translation, which are based on domain-specific translation lexicon (Resnik and Melamed 1997; Hiemstra et al. 1997; Peñas et al. 2012). However, developing a domain-specific dictionary requires a significant effort. In this paper, we propose a new approach to support domain-specific translation. We analyze a corpus of 30,000 Stack Overflow questions to build a domain-specific vocabulary based on the term frequency of each English word. The corpus contains a total of 111,174 English words.

Given a Chinese word, our *CLRQR* system first uses a general domain translation tool (e.g., Youdao translation API⁵) to obtain a few translation candidates. Then, based on the term frequency of the translation candidates in the domain-specific vocabulary, it selects those words whose term frequency are greater than the mean frequency as the translation. According to practical experience, we find that the performance of selecting words whose term frequency are greater than the mean frequency is always better than selecting the only word with highest term frequency. For example, for the Chinese word “方法”, the term frequency of the translation “way”, “method”, “mean” and “function” is 2,405, 60,960, 5,307, 125,862 respectively. As a result, “function” and “method” are selected as the translation of the Chinese word “方法” which are widely used in Java. In fact, the two translations are considered as “synonyms” as many developers always mix these two words in their discussions.⁶ This approach has two advantages over domain-specific dictionary: 1) it is based on

³FudanNLP, available at <http://nlp.fudan.edu.cn>

⁴IctclasNLP, available at <http://ictclas.nlpir.org/docs>

⁵Youdao translation API, available at <http://fanyi.youdao.com/openapi>

⁶Java’s methods versus functions, available at <http://stackoverflow.com/questions/16223531/javas-methods-vs-functions>

Search

1 result

relevance

newest

votes

active

0

votes

1

answer

Q: Joda Time sometimes returns wrong time

"); DateTimeZone dateTimeZone = DateTimeZone.forTimeZone(timeZone); DateTime dateTime = new DateTime(dateTimeZone); So **sometimes** the **time** comes out correct but **sometimes** it is tipped one hour ... Whenever I pass **time** zone to create a DateTimeZone and the get the **time** out of it, the **time** returned is fluctuating. Here is my code: TimeZone timeZone = TimeZone.getTimeZone("America/Belize ...

java

datetime

jodatime

asked 26 mins ago by [john.p.doe](#)

Fig. 3 Search “*Joda Time sometimes returns wrong time*” on Stack Overflow

the general domain translation tool; and 2) it utilizes the crowdsourced knowledge to build a domain-specific vocabulary with a very small effort.

2.2.3 Challenges in Question Retrieval Algorithm

When searching on Stack Overflow, we find that the question retrieval algorithm is not very robust. For example, when we search the question “*Joda Time sometimes returns wrong time*”, we can retrieve a question on Stack Overflow successfully as shown in Fig. 3. However, if we change the word “returns” to “return”, Fig. 4 shows that the search for “*Joda Time sometimes return wrong time*” returns no matches. It seems that Stack Overflow question retrieval algorithm does not take word stemming into consideration. Furthermore, we observe that keywords extracted from different parts of the question (such as title versus description) often have different levels of importance for question retrieval. However, existing question retrieval algorithms do not take this into account. In this paper, we design a question retrieval algorithm to address these limitations by considering word stemming and assigning different weights to the words from title and description. Furthermore, we

Search

0 results

relevance

newest

votes

active

Your search returned no matches.

Fig. 4 Search “*Joda Time sometimes return wrong time*” on Stack Overflow

propose three different similarity metrics to determine the relevance between query and questions.

3 The Approach

In this section, we first present the overall framework of our proposed approach to cross-language question retrieval (Section 3.1). Then, we describe the details of essential information extraction, domain-specific cross-language translation, and question retrieval algorithms in Sections 3.2, 3.3 and 3.4, respectively.

3.1 Overall Framework

Figure 5 presents the overall framework of our domain-specific cross-language relevant question retrieval. Given a software engineering related question in Chinese, essential information extraction (step 1) extracts Chinese words and English words from the given Chinese question. Given the extracted Chinese words, domain-specific cross-language translation (step 3) translates the Chinese words into domain-specific English words, based on a domain-specific vocabulary derived from a corpus of Stack Overflow questions (step 2). Given a list of candidate English words (extracted or translated) from the input Chinese question, the system formulates an English query as follows: first, it stems the English words (step 4), then it assigns different weights to the different types of English words (step 5) depending on whether the words are from the question title or description, and finally it takes a subset of English words with the highest scores to formulate the English query (step 6). The system uses the English query to search a repository of Stack Overflow questions (step 7), and the retrieval algorithms using different similarity metrics return the top-10 most relevant English questions to the user.

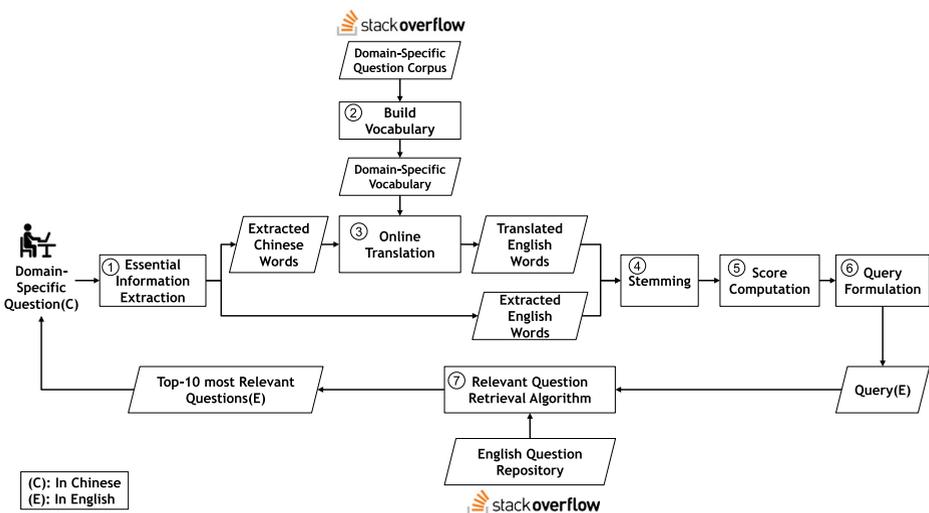


Fig. 5 The Framework of Domain-Specific Cross-Language Question Retrieval

Algorithm 1 Chinese Keyword Extraction

```
1: Input:  
2: Sentence: The question description  
3: Output:  
4: CKL: List of Chinese keywords in the Sentence  
5: Method:  
6: KeywordsList1 = FudanNLP(Sentence);  
7: for all Keywords  $KW^i \in \textit{KeywordsList1}$  do  
8:   CKL.Add( $KW^i$ );  
9: end for  
10: KeywordsList2 = IctclasNLP(Sentence);  
11: for all Keywords  $KW^i \in \textit{KeywordsList2}$   
12:   CKL.Add( $KW^i$ );  
13: end for  
14: Output CKL;
```

3.2 Essential Information Extraction

Given a question in Chinese, we extract its essential information based on the following two observations:

1. Question title summarizes the core issue of the question, while the question description contains the details of the question. Information in question title is generally more important than that in question description.
2. Chinese questions contain a mix of Chinese and English words. Most English words are technical words that are important for retrieving relevant English questions.

Therefore, we have two dimensions for essential information extraction: the location of the information (question title or description) and the language of the information (Chinese or English). Combining the two dimensions, we divide the essential information into four kinds: (i) Chinese words in Title (*CT*), (ii) English words in Title (*ET*), (iii) Chinese Keywords in Description (*CKD*), and (iv) English words in Description (*ED*). As described in Section 3.4.2, we assign different weights to different kinds of essential information for measuring their importance for relevant question retrieval.

Given a Chinese question, we first extract all Chinese words in Title, all English words in Title and all English words in Description as essential information. Algorithm 1 presents the implementation of our composite Chinese keyword extraction algorithm for extracting Chinese keywords in Description. The input is the question description. The Algorithm 1 uses two different popular Chinese keyword extraction algorithms (FudanNLP & IctclasNLP). FudanNLP is based on TextRank algorithm. TextRank is a graph-based ranking model for ranking important text units (e.g., words, phrases or sentences) in a document or corpus (Mihalcea and Tarau 2004). For keyword extraction, FudanNLP creates a graph of the words and the co-occurrence relationships between words within windows of words from a document. It then identifies the most important vertices of the graph (words) based on a random walk of the word graph. IctclasNLP is based on entropy. It uses the cross information entropy to compute the context entropy of each candidate word. The words with higher context entropy are more likely to be the keywords. The two algorithms produce two different but complementary sets of Chinese keywords. To reduce the bias caused by a single

method, the algorithm takes the union of the two keyword sets as the final set of Chinese keywords in Description.

For the motivating example shown in Fig. 1, the system extracts three Chinese words from the title: “开源”, “审查” and “工具”, and five Chinese keywords from the description: “项目”, “开源”, “代码”, “审查” and “工具”. There are no English words in the title of this question. The system extracts two English words from the question description: “java” and “javaweb”.

For questions of a particular domain (e.g., Java), we consider the domain word (e.g., “java”) not an important word to distinguish the core issues of different questions. Therefore, we discard the English word “java” and keep only the English word “javaweb”.

3.3 Domain-Specific Cross-Language Translation

Translation is a critical step in cross-language information retrieval (Kraaij et al. 2003; Aceves-Pérez et al. 2007). Recent results show that the challenge lies in how to differentiate a word’s domain-specific meaning from its common meaning. General domain translation tools like Youdao Translation and Google Translation, may not perform well for domain-specific translation, because it does not consider any domain knowledge. For a given Chinese word, general domain translation tools usually return two types of translation results: a basic translation result and several web translation results. Basic translation result means the most common translation result. Web translation result means the results are extracted from many web pages based on some NLP techniques.⁷ However, neither the basic translation nor the web translation results consider domain-specific meanings of translations.

We proposed a method to address the lack of domain-specific meanings of general translations. Figure 6 presents the details of our method. We divide the domain-specific cross-language translation into an offline vocabulary building step and an online translation step. To build a domain-specific vocabulary, we make use of crowdsourced knowledge in Stack Overflow discussions. Take the domain Java as an example. We collect a corpus of randomly-selected 30,000 Stack Overflow questions tagged with *java*. Sufficient questions have to be collected in order to obtain a vocabulary that reaches a high coverage of English words in questions of a particular domain (see Section 4.2 for the vocabulary coverage analysis in our experimental questions). We first remove stop words (step 1), such as “hello”, “the” and “you”. The stop-word list we use is available at Snowball.⁸ Then, we compute term frequency for each word in the corpus and build a vocabulary of each word and their term frequency in the corpus (step 2). The resulting vocabulary contains a total of 111,174 English words. The same process can be adopted to build domain-specific vocabularies for other domains like Python or .NET.

For online translation, given a Chinese word (which can be a Chinese word in Title or a Chinese keyword in Description), the system first uses a general Chinese-English translation tool (Youdao translation API is used in this work) to obtain a list of candidate English words which includes both basic translation result and several web translation results (step 3). Then, the system checks the term frequency of these English word candidates in the domain-specific vocabulary (step 4). Finally, the system selects the English words with the term frequency above the mean term frequency of all the candidate English words as the

⁷Web translation result, <http://faq.youdao.com/dict/?p=65>

⁸Stop-word list, available at <http://snowball.tartarus.org/algorithms/english/stop.txt>

Offline: Build Domain-Specific Vocabulary

Online: Domain-Specific Translation

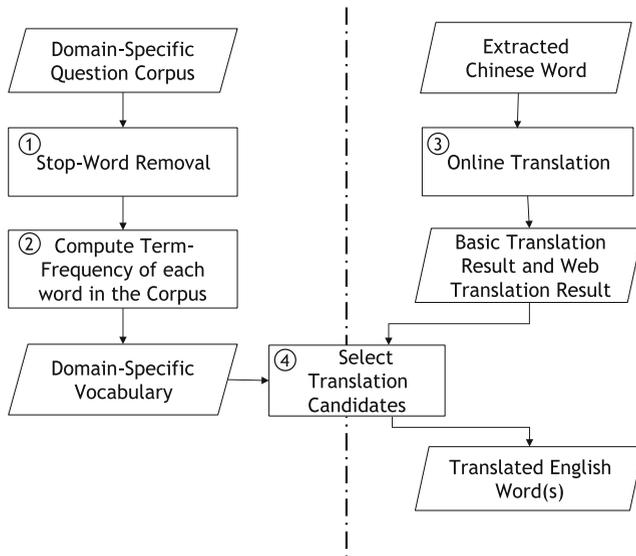


Fig. 6 Domain-Specific Cross-Language Translation

translation of the given Chinese word. Although there can be potentially several English translations for a Chinese word, we find that most Chinese words end up with only one domain-specific English translation in practice (see Section 4.2). If none of the candidate English words exist in the domain-specific vocabulary, the system returns the basic translation (i.e., the most common translation) as the translation result.

Table 2 shows the general and final translation results of the Chinese words extracted from the Chinese question in Fig. 1. In particular, the English words “project”, “opensource”, “code”, “investigate” and “tool” are the basic translation results for the corresponding Chinese words. Our method returns the English words “project”, “opensource”, “code”, “review”, “tool” as the final translation of the Chinese word

Table 2 Examples of Domain-specific Cross-language Translation

Chinese Word	Translation results	Term Frequency in Vocabulary	Mean	Final Result
项目	project(basic) item article	21,021 2,642 3,101	8,921	project
开源	opensource(basic)	124	124	opensource
代码	code(basic) word	408,565 9,932	209,249	code
审查	investigate(basic) review examine	1,857 2,589 1,807	2,084	review
工具	tool(basic) instrument	7,481 2,515	4,998	tool

“项目”，“开源”，“代码”，“审查” 和 “工具”。Note that for “审查”，the translation “review” is a better domain-specific translation than the basic translation “investigate”.

To demonstrate the usefulness of domain-specific translation for question retrieval, we build a baseline in our evaluation. Given a Chinese word, the baseline approach does not do any domain-specific translation. Instead, it simply uses the basic translation as the translation result of the Chinese word.

3.4 Relevant Question Retrieval Algorithm

Our cross-language question retrieval contains three steps: stemming, word score computation and query formulation, and question retrieval.

3.4.1 Stemming

In the stemming step, we reduce each word to its root form, for example, words “write” and “written” are both reduced to “writ”. In this work, we use a popular stemming algorithm (the Porter stemmer (Porter 1980)).

3.4.2 Word Score Computation

Considering different contributions of a word to express the core issue of the question, we assign different weights to each kind of words depending on their location and language.

We observe that the question title usually sums up the core issue of the question better than the question description. Therefore, we set the weight of the words in the title higher than the words in the description. However, the question description usually contains more technical details than the question title. Setting a too high weight for the words in the title will overwhelm the technical words in the question description. A Chinese question usually mixes the Chinese words and English words. We observe that both Chinese words and English words are important for question retrieval.

Therefore, we set the same weight to the English words and the Chinese words.

As a result, Chinese words in Title (*CT*) and English words (*ET*) in Title have one weight, and Chinese keywords in Description (*CKD*) and English words in Description (*ED*) have another weight. In this work, we set the weight of the words in the title three times higher than the words in the description, i.e., *CT:ET:CKD:ED* as 3:3:1:1. We also experiment other weight settings in the RQ4 in Section 5.

For each English word (either original English words extracted from question title and description or the translated English words using our domain-specific translation method), we update its *Score* by $(termfrequency \times Weight) / WordsetSize$.

If a word belongs to different kinds of word set at the same time, the score of the word will be accumulated. Table 3 shows the score computation for the words in the question shown in Fig. 1. For example, the word “tool” (the translation for the Chinese word “工具”) is both a Chinese word in Title and a Chinese keyword in Description. For the Chinese words in Title, we calculate the score of the word “tool” as 1, i.e., $(1 \times 3) / 3 = 1.00$. For the Chinese keywords in Description, we calculate the score of the word “tool” as 0.20, i.e., $(1 \times 1) / 5 = 0.20$. Thus, we get the final score for the word “tool” as 1.20.

According to practical experience in web log analysis (Liu et al. 2002; Cui et al. 2002), we use up to six keywords to express the core issue of a question. After computing the score of all the words, we select up-to six words with the highest scores to formulate an English query. In this example, the English query contains six query words: “code”, “review”,

Table 3 Different Scores for 4 Types of Words

Location	Word language	Weight	WordsetSize	Score (desc)
Title	English Words ()	3	0	code : 1.20 review : 1.20
	Chinese Words (code, review, tool)	3	3	tool : 1.20 javaweb : 1.00
Description	English Words (javaweb)	1	1	project : 0.20 opensource : 0.20
	Chinese Keywords (tool, code, review, project, opensource)	1	5	

“tool”, “javaweb”, “project” and “opensource”. If there are two or more English words with same score at the cut-off point (i.e., rank 6), our approach will randomly select one of the equal-score words.

3.4.3 Relevant Question Retrieval

Question retrieval algorithms calculate the relevance between the query words and each English question in the repository, and recommend the top-10 most relevant English questions to users that may help them solve the problem in the given Chinese question.

We propose three different retrieval algorithms (i.e., $CLRQR_{WM}$, $CLRQR_{VSM}$ and $CLRQR$) based on different heuristics, word representations and similarity metrics. We compare the effectiveness of different algorithms in the evaluation to study which one is the most suitable for the question retrieval task.

- **$CLRQR_{WM}$, a word matching based retrieval algorithm.** The typical approach to measuring relevance between a set of query words and each English question in a repository is to use a simple lexical matching method, and produce a similarity score based on the number of lexical units that appear in both the input query word set and the text of each English question. Improvements to this simple method have considered stemming, stop-word removal, part-of-speech tagging, longest subsequence matching, as well as various weighting and normalization factors (Salton and Buckley 1988).

Given a query word with its corresponding weight, the algorithm $CLRQR_{WM}$ (i.e., Algorithm 2) firstly calculates the term frequency of word in the title and description of each English question in the question repository (lines 16-17). Then, it computes the relevance between a query word (Q_w) and an English question (Q) by the (1) (line 18). Due to the observation in Section 3.2, we consider that the word in Q_wMap from the title of an English question is more important than those from the question description. Thus, we set different weights to the words from the title and the description (i.e., 2 and 1 respectively). To reduce the bias caused by a word with too high term frequency, the algorithm then counts the number of times (i.e., *ContainNum*) that a word appears in the title and description of the English question, and divide the times by the total number of query words (i.e., the size of Q_wMap) to have contain-words-ratio (*ContainRatio*). The algorithm sums the multiplication of the contain-words-ratio and the query-question relevance for all the query words following the (2) (line 21). The

final query-question relevance score is used to rank the questions to be returned (line 24).

Algorithm 2 $CLRQR_{WM}$ Relevant Question Retrieval Algorithm

```

1: Input:
2:  $QwMap$ : Map<Query word, Score>
3:  $QEQL$ : Domain-specific English Question List
4: Output:
5:  $TRQ$ : Top-10 most Relevant Questions
6: Method:
7: Map<Question, Relevance>  $QSMAP = NULL$ ;
8:  $SIZE = QwMap.size()$ ;
9: for all Question  $Q^i \in QEQL$  do
10:    $Relevance = 0$ ;
11:    $ContainNum = 0$ ;
12:   for all Entry<Query word, Score>  $Qw^i \in QwMap$  do
13:     if  $Q^i.title.contains(Qw^i)$  or  $Q^i.desc.contains(Qw^i)$  then
14:        $ContainNum++$ ;
15:     end if
16:      $tf\_Title = calcTF(Q^i.title, Qw^i)$ ;
17:      $tf\_Desc = calcTF(Q^i.desc, Qw^i)$ ;
18:      $Relevance += (tf\_Title*2 + tf\_Desc) * (Qw^i.Score)$ ;
19:   end for
20:    $ContainRatio = ContainNum / SIZE$ ;
21:    $Relevance *= (ContainRatio)$ ;
22:    $QSMAP.put(Q^i, Relevance)$ ;
23: end for
24:  $TRQ = getTop-10Questions(QSMAP)$ ;
25: Output  $TRQ$ ;

```

$$Relevance(Qw, Q) = (tf_Q.Title \times 2 + tf_Q.Desc) \times Qw.score \quad (1)$$

$$Relevance(Query, Q) = \sum_{i=1}^M Relevance(Qw, Q) \times ContainRatio \quad (2)$$

- $CLRQR_{VSM}$, a vector space model based retrieval algorithm. $CLRQR_{VSM}$ represents the query and the questions as term vectors. When text is represented as term vectors, the correlation between the text vectors reflects the similarity between documents. The correlation between vectors is often quantified as the cosine of the angle between vectors, that is, the cosine similarity. Cosine similarity is one of the most popular similarity measure applied to text documents, such as in numerous information retrieval applications (Baeza-Yates et al. 1999) and data mining (Tan and et al 2006).

Given a query of n words, the $CLRQR_{VSM}$ algorithm refers to each word in the query as one dimension. Then, the vector of the given query (i.e., $vector_{Query}$) can be obtained as:

$$vector_{Query} = \left\langle \frac{Score_1}{Sum}, \frac{Score_2}{Sum}, \dots, \frac{Score_n}{Sum} \right\rangle$$

where Sum equals to $\sum_{i=1}^n Score_i$ and $\frac{Score_i}{Sum}$ ($1 \leq i \leq n$) is the word score of the i th word in the query.

For a given English question in a repository, the algorithm firstly calculates term frequency of the query word in the title and description of the question. The more frequently a word appears, the more important it is. Same as $CLRQR_{WM}$, the algorithm $CLRQR_{VSM}$ also considers that the query word from the title of an English question is more important than those from the question description. Thus, two factors are considered as a measure of the importance of a query word in a given English question: term frequency and appearing in the title or description. Then, the vector of an English question (i.e., $vector_Q$) can be obtained as follow:

$$vector_Q = \left\langle tf_1^{Title} * 2 + tf_1^{Desc}, tf_2^{Title} * 2 + tf_2^{Desc}, \dots, tf_n^{Title} * 2 + tf_n^{Desc} \right\rangle$$

In the above vector, tf_i^{Title} refers to the term frequency of the i -th query word in Title and tf_i^{Desc} refers to the term frequency of i -th query word in Description.

Given two vectors, A and B , the cosine similarity is represented using a dot product and magnitude as:

$$CosineSim(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \tag{3}$$

The resulting similarity ranges from -1 meaning exactly opposite, to 1 meaning exactly the same, with 0 indicating orthogonality (decorrelation) and in-between values indicating intermediate similarity. Thus, the relevance between the query and each question in the repository is defined as follow:

$$Relevance(Query, Q) = \frac{1}{|1 - CosineSim(vector^{Query}, vector^Q)|} \tag{4}$$

- CLRQR, a word embedding based retrieval algorithm.** $CLRQR$ exploits distributed word representation, i.e., word embedding. Distributed word representations assume that words appear in similar context tend to have similar meanings (Harris 1954). Therefore, individual words are no longer treated as unique symbols, but mapped to a dense real-valued low-dimensional vector space. Each dimension represents a latent semantic or syntactic feature of the word. Semantically similar words are close in the embedding space. Figure 7 shows some examples of domain-specific word embeddings we learn from the text of Stack Overflow *java* questions, visualized in a two-dimensional space using the t-SNE dimensionality reduction technique (Maaten and Hinton 2008). Semantically close words, such as *JPanel*, *JButton*, *JFrame* and *JLabel* which are GUI components are close in the vector space.

Word embeddings are typically learned using neural language models. In this work, we use a continuous skip-gram model, a popular word to vector (word2vec) model proposed by Mikolov et al. (2013a, b). Continuous skip-gram model learns the word embedding of a center word (i.e., w_i) that is good at predicting the surrounding words in a context window of $2k + 1$ words ($k = 2$ in this example). The objective function of

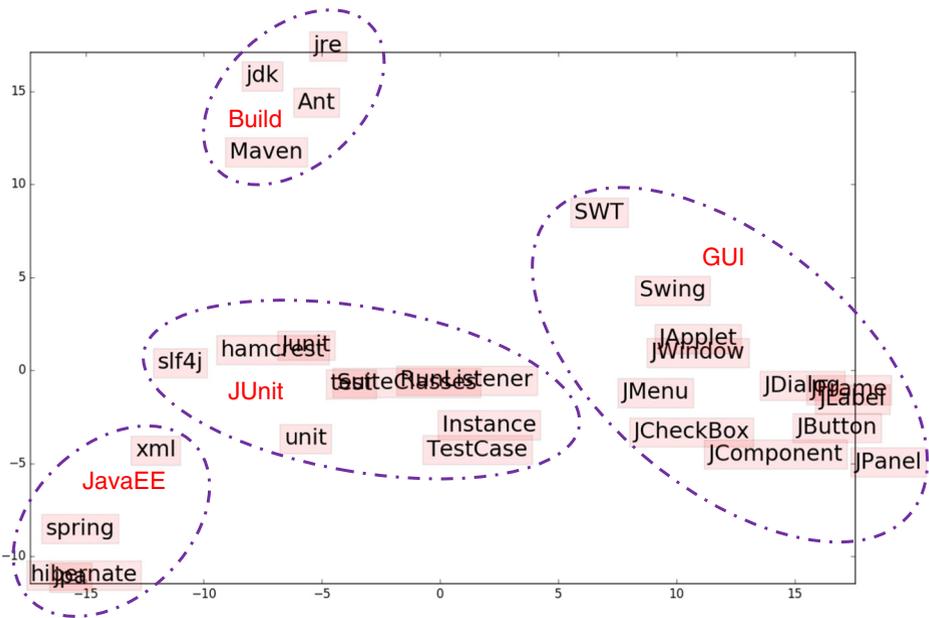


Fig. 7 Example of Word Embedding

skip-gram model is to maximize the sum of log probabilities of the surrounding context words conditioned on the center word:

$$\sum_{i=1}^n \sum_{-k \leq j \leq k, j \neq 0} \log p(w_{i+j}|w_i) \quad (5)$$

where w_i and w_{i+j} denote the center word and the context word in a context window of length $2k + 1$. n denotes the length of the word sequence.

The $\log p(w_{i+j}|w_i)$ is the conditional probability defined using the *softmax function*:

$$p(w_{i+j}|w_i) = \frac{\exp(v_{w_{i+j}}^T v_{w_i})}{\sum_{w \in W} \exp(v_w^T v_{w_i})} \quad (6)$$

where v_w and v'_w are respectively the input and output vectors of a word w in the underlying neural model, and W is the vocabulary of all words. Intuitively, $p(w_{i+j}|w_i)$ 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. Mikolov et al. (2013b) propose an efficient negative sampling method to compute this probability. The output of the model is a dictionary of words, each of which is associated with a vector representation. Thus, given a word, it will be converted into a high dimensional vector with real value by looking up the dictionary of word embeddings.

Given a query word vector $vector_{Q_w}$ and a word vector $vector_{q_w}$ in English question, we define their semantic similarity as the cosine similarity between their learned word embeddings:

$$Relevance(vector_{Q_w}, vector_{q_w}) = \text{CosineSim}(vector_{Q_w}, vector_{q_w}) \quad (7)$$

This is simply the inner product of the two vectors, normalized by their Euclidean norm. To compute the relevance between the query and the question text, we use the text-to-text similarity measure introduced by Mihalcea et al. (2006). According to Mihalcea et al. (2006), the relevance between a query word Qw and an English question Q is computed as the maximum similarity between Qw and any word w' in Q :

$$Relevance(Qw, Q) = \max_{w' \in Q} Relevance(Qw, w') \times Qw.score \quad (8)$$

Then, we define the relevance between a given query $Query$ and an English question Q in a repository as:

$$Relevance(Query, Q) = \sum_{Qw \in Query} Relevance(Qw, Q) \quad (9)$$

4 Experimental Methods

This section describes the experiment design for evaluating our approach.

4.1 Research Questions

Our evaluation aims to answer the following five research questions:

- RQ1: *How effective are the three proposed retrieval algorithms (i.e., word matching, vector space model, and word embeddings)? Which retrieval algorithm is most suitable for our task?*
- RQ2: *How effective is our domain specific translation method? How much improvement can it achieve over the baseline approach which directly uses the basic translation result?*
- RQ3: *How effective is our approach in cross-language relevant question retrieval? How much improvement can it achieve over the baseline approaches?*
- RQ4: *What is the impact of different weight settings for different kinds of essential information on the performance of question retrieval?*
- RQ5: *What is the time efficiency of our system?*

4.2 Dataset

To demonstrate the generality of our approach for different domains, we test our approach for Java, Python and .NET questions. We prepare query Chinese questions, English questions repositories and word embedding corpora as follows.

Query Chinese Questions We crawl all the Chinese technical questions from SegmentFault and V2EX and randomly select 80 Java, 20 Python and 20 .NET questions as query Chinese questions to evaluate our approach. Table 4 shows the distribution of query Chinese questions from SegmentFault and V2EX, respectively. Since V2EX does not have .NET questions, the 20 .NET Chinese questions are all from SegmentFault. We manually check the selected questions and find that those questions are diverse and cover different aspects of Java, Python or .NET technologies.⁹ Meanwhile, the number of questions is manageable

⁹The 120 Query Chinese questions, available at <https://goo.gl/zAbLVp>

Table 4 Distribution of Query Chinese Questions from *SegmentFault* and *V2EX*

	SegmentFault	V2EX	Total
Java	40	40	80
Python	10	10	20
.NET	20	NA	20

for the manual inspection of the question retrieval results which is labor intensive and time consuming.

Table 5 presents the number of English words in the 120 query Chinese questions and the number of English technical words among these English words. We can see that 1775 (96.2%) of the 1845 English words in these Chinese questions are technical words. This confirms our observation that most English words in Chinese questions are actually technical words.

English Questions Repositories We extract English questions from Stack Exchange Data Dump¹⁰ of March 2016. Specifically, we extract all 714,599, 715,119 and 248,853 English questions which are tagged as “java”, “python” and “.net” as English question repositories for Java, Python, and .NET, respectively.

We build domain-specific vocabularies for Java, Python and .NET as described in Section 3.3. There are 111,174, 69,722 and 29,367 different English words in the Java, Python and .NET vocabularies, respectively. For each of the 120 query Chinese questions, there are on average 6.15 Chinese words that need to be translated into English. Each Chinese word has an average of 2.7 candidate English translation results and 2.5 of them exist in the corresponding domain-specific vocabulary. That is, our domain-specific vocabulary has a high coverage of the English translation of Chinese keywords of the randomly selected query Chinese questions. After selecting domain-specific translation using the domain-specific vocabulary, on average 1.14 of candidate English words are selected. That is, 1.56 of candidate English words are filtered by the domain-specific vocabulary. This indicates that in most cases a Chinese word will be translated into one domain-specific English word. Therefore, the impact of false positives of query expansion should be minor.

Word Embedding Corpora We experiment with using several word embedding corpora of different sizes (with the increments of 100,000 questions) to train word embeddings. We find that the impact of the size of a corpora on the corpora vocabulary and the resulting word embeddings is minor. In the end, we randomly select 300,000, 300,000 and 200,000 English questions from the English questions repositories for Java, Python and .NET as the word embedding corpora for Java, Python and .NET, respectively. Each corpora contains a total of 329,845, 720,423 and 438,686 English words respectively. We train the word embeddings using the word2vec tool Gensim (Rehůrek and Sojka 2010).

4.3 Baseline Building

Table 6 summarizes different approaches used for answering the research questions RQ1, RQ2 and RQ3. In our experiment, all the approaches return the top-10 most relevant English questions for each query Chinese question. In the following discussion, *CLRQR*

¹⁰Stack Exchange Data Dump, available at <https://archive.org/download/stackexchange>

Table 5 Statistics of English Words and English Technical Words in the 120 Query Chinese Questions

Domain	#Question	#English words	#English technical words
Java	80	1002	993
Python	20	516	477
.NET	20	327	305
#Total	120	1845	1775(96.2%)

represents the default cross-language question retrieval approach. It uses word-embedding based retrieval algorithm. We refer to *CLRQR* as “our approach”, unless otherwise stated.

For RQ1, we compare *CLRQR* with the two variants of *CLRQR* that use word-matching and vector-space-model based retrieval algorithm, respectively. These two variants are referred to as *CLRQR_{wm}* and *CLRQR_{vsm}*. *CLRQR*, *CLRQR_{wm}* and *CLRQR_{vsm}* differ only in the retrieval algorithm.

For RQ2, we compare *CLRQR* with another variant of *CLRQR* which is referred to as *CLRQR^{BT}*. *CLRQR* uses our domain-specific translation method, while *CLRQR^{BT}* directly use the basic translation result. Other than that, *CLRQR* and *CLRQR^{BT}* are the same.

For RQ3, we compare *CLRQR* with four baseline approaches. These four baseline approaches are designed to combine different sources of query words, query formulation mechanisms and search engines. In the translation phase, *BaselineApproach1* and *BaselineApproach3* translate only Chinese words in Title, while *BaselineApproach2* and *BaselineApproach4* translate Chinese words in both Title and Description. All the baseline approaches use the Google translation. After the translation, we also remove English stop words from the translated words which is the same as our approach. We refer the remaining English words (originally in the Chinese question or translated from the Chinese words) as the “Query Words”.

Stack Overflow search engine and Google search engine are the two widely used search engines for relevant question retrieval.

BaselineApproach1 and *BaselineApproach2* use Stack Overflow search engine, while *BaselineApproach3* and *BaselineApproach4* use Google search engine. After removing the stop words from the question title and the question description, we formulate the query for the baseline approaches from the first word of the question title and description up to the length limits of the query that a search engine allows. Different search engines have different limits for the length of a query. For the Stack Overflow search engine, the query cannot exceed 140 characters. If the length of the question title and description exceeds 140 characters, we keep the complete words up to the largest character index below or equal to 140. Table 7 presents an example. As the word “type” is across the limit 140, we keep the words before the word “type” as the query for the Stack Overflow search engine. For the Google search engine, the query cannot exceed 32 words. Thus, we only keep the first 32 words as query if the question title and description contain more than 32 words.

Furthermore, in order to make fair comparison, we optimize the search scope for baseline approaches based on their corresponding search engine. When formulating the query, *BaselineApproach1* and *BaselineApproach2* append “[domain] is:question” to the query, which instructs the Stack Overflow search engine search only questions tagged with the domain word like *Java*. *BaselineApproach3* and *BaselineApproach4* append “site:StackOverflow.com” to the query, which instructs the Google search engine to search only the Stack Overflow web site.

Table 6 Approaches for Cross-language Relevant Question Retrieval

Research question	Approach name	Translation item	Translation result selection	Query formulation	Retrieval algorithm or search engine
ALL	<i>CLRQR</i>	Title and Description	Filtered by Domain-Specific Vocabulary	Scored Query Words	Word Embedding
RQ1	<i>CLRQR_{WM}</i>				Word Matching
	<i>CLRQR_{VSM}</i>		Directly use the basic translation result		Vector Space Model
	<i>CLRQR_{BT}</i>		Google Translate		Word Embedding
RQ3	<i>BaselineApproach1</i>	Title Only		“[domain] is:question” + “Title Query Words”	Stack Overflow Search Engine (not exceeding 140 characters)
	<i>BaselineApproach2</i>	Title and Description		“[domain] is:question” + “Title Query Words” + “Description Query Words”	
	<i>BaselineApproach3</i>	Title Only		“Title Query Words” + “site:StackOverflow.com”	Google Search Engine (not exceeding 32 words)
	<i>BaselineApproach4</i>	Title and Description		“Title Query Words” + “Description Query Words” + “site:StackOverflow.com”	

Table 7 Example for Query Formulation of the Baseline Approaches using the Stack Overflow Search Engine

Index	1	2	3	...	137	138	140	141	142
Query	S	p	r	...	<space>	t	y	p	e

4.4 User Study

We conduct a user study to evaluate the top-10 most relevant questions returned by different approaches.

Participants We recruited participants through our school's mailing lists and selected five master students for each domain of questions (Java, Python, .NET) to join our user study. That is, we have three groups and each group includes five different participants (i.e., 15 participants in total). These participants are not affiliated with our research project. All the participants have industrial experience in corresponding programming domain (ranging from 3 to 6 years). All the participants have passed the national college English test (Level 4).

Procedure We provided the participants query Chinese questions from SegmentFault and V2EX. For each query Chinese question, we provided a list of the top-10 most relevant English questions generated by different approaches (see Table 6). The participants did not know which result is generated by which approach.

We asked the participants to read the same question at the same time and independently evaluate whether each retrieved English question is relevant or not to the given Chinese question. If more than half of the participants consider an English question as relevant to the given Chinese question, then this English question will be recorded as an actually relevant English question. To minimize the impact of fatigue, we divided our user study into six sessions – 20 query Chinese questions is analyzed in each session.

Each session took about 2-3 hours.

4.5 Evaluation Metrics

We use the following metrics to compare the baseline approaches and our approach (Xia and Lo 2017; Zhou et al. 2012; Bao et al. 2017; Xia et al. 2015):

- **Precision@K.** Precision is the percentage of actually relevant questions out of the questions that a relevant question retrieval approach returns for a Chinese question. It is defined as:

$$Precision@k = \frac{ARQs \text{ in top-}k}{k} \quad (10)$$

In the above equation, *ARQs* refers to actually relevant questions. In this paper, we set $k = 1, 5$ and 10 .

- **Recall@K.** Recall is the percentage of actually relevant questions returned for a Chinese question out of all the relevant questions in the repository. It is defined as:

$$Recall@k = \frac{ARQs \text{ in top-}k}{ARQs \text{ in Repository}} \quad (11)$$

It is impractical to check every question in repository to determine its relevance to the query Chinese question. Therefore, *ARQs in Repository* refers to the union of

actually relevant questions returned by all approaches for a Chinese question. In this paper, we set $k = 1, 5$ and 10 .

- **Top-K Accuracy.** Top-k accuracy is the percentage of Chinese questions for which at least one actually relevant question is ranked within the top-k position in the returned lists of English questions. Given a Chinese question CQ , if at least one of the top-k most relevant English questions is actually relevant, we consider the retrieval to be successful, and set the value $Success(CQ, top - k)$ to 1; else we consider the retrieval to be unsuccessful, and set the value $success(CQ, top - k)$ to 0. Given a set of Chinese questions, denoted as CQs , its top-k accuracy Top@k is computed as:

$$Top@k = \frac{\sum_{CQ \in CQs} Success(CQ, top-k)}{|CQs|} \quad (12)$$

The higher the top-k accuracy score is, the better a relevant question retrieval approach performs. In this paper, we set $k = 1, 5$ and 10 .

- **Mean Reciprocal Rank (MRR).** MRR is a popular metric used to evaluate an information retrieval technique (Baeza-Yates et al. 1999). Given a query (in our case: a Chinese technique question), its reciprocal rank is the multiplicative inverse of the rank of the first correct document (in our case: actually relevant English question) in a rank list produced by a ranking technique (in our case: relevant question retrieval approach). Mean Reciprocal Rank (MRR) is the average of the reciprocal ranks of all Chinese questions in a set of Chinese questions. The MRR of a set of Chinese questions is computed as:

$$MRR(R) = \frac{1}{|CQs|} \sum_{CQ \in CQs} \frac{1}{Rank(CQ)} \quad (13)$$

In the above equation, $Rank(CQ)$ refers to the position of the first actually relevant English question in the ranked list of English questions returned by a cross-language relevant question retrieval approach for a Chinese question.

- **Mean Average Precision (MAP).** MAP is a single-figure measure of quality, and it has been shown to have especially good discrimination and stability to evaluate ranking techniques (Baeza-Yates et al. 1999). Different from top-k accuracy and MRR that only consider the first correct result, MAP considers all correct results. For a query (in our case: a Chinese technique question), its *average precision* is defined as the mean of the precision values obtained for different sets of top k documents (in our case: actually relevant English questions) that were retrieved before each relevant document is retrieved, which is computed as:

$$AvgP(CQ) = \frac{\sum_{j=1}^M P(j) \times Rel(j)}{ARQs \text{ in Repository}} \quad (14)$$

In the above equation, M is the number of English questions in a ranked list, $Rel(j)$ indicates whether the English question at position j is actually relevant or not (in our case: the question is judged by the users as actually relevant or not), and $P(j)$ is the precision at the given cut-off position j and is computed as:

$$P(j) = \frac{ARQs \text{ in top } j \text{ positions}}{j} \quad (15)$$

Then the MAP for a set of Chinese questions CQs is the mean of the average precision scores for all Chinese questions in CQs :

$$MAP = \frac{\sum_{CQ \in CQs} AvgP(CQ)}{|CQs|} \quad (16)$$

In relevant question retrieval, a Chinese question may have a number of relevant English questions. We use MAP to measure the average performance of different approaches to retrieve all of the relevant questions. The higher the MAP value, the better the relevant question retrieval approach performs.

5 Experimental Results

In our experiment, we are interested in the following five research questions:

RQ1: *How effective are the three proposed retrieval algorithms (i.e., word matching, vector space model and word embeddings)? Which retrieval algorithm is most suitable for our task?*

Motivation Three different relevant question retrieval algorithms are proposed in this work. These algorithms adopt different word/document representations and similarity metrics. We would like to investigate whether these different word/document representations and similarity metrics may result in different retrieval performance, and if so, which algorithm is most suitable for the task.

Approach We compare the performance of the three algorithms (namely $CLRQR$ using word embeddings, $CLRQR_{wm}$ using word matching, and $CLRQR_{vsm}$ using vector space model) in terms of the Precision@k, Recall@k, Top-k accuracies (k = 1, 5 and 10), MRR, and MAP. Moreover, we apply the Wilcoxon signed-rank test (Wilcoxon 1945) at 95% significance level on the paired data which corresponds to the precision@k, recall@k, top-k accuracies, MRR, and MAP scores of the best algorithm and the second best algorithm to test the statistical significance between the performance difference.

Result Table 8 presents the Precision@k, Recall@k, Top-k accuracies (k=1, 5 and 10), MRR and MAP for $CLRQR$, $CLRQR_{WM}$ and $CLRQR_{VSM}$, respectively. We notice that $CLRQR$ achieves much better performance in all evaluated metrics by a substantial margin,

Table 8 Precision@k, Recall@k, Top-k Accuracies (k=1, 5 and 10), MRR and MAP for $CLRQR$, $CLRQR_{WM}$ and $CLRQR_{VSM}$

Approach	Precision@1	Precision@5	Precision@10	Recall@1	Recall@5	Recall@10
$CLRQR$	0.81	0.77	0.72	0.05	0.20	0.36
$CLRQR_{WM}$	0.61	0.46	0.37	0.03	0.11	0.17
$CLRQR_{VSM}$	0.32	0.30	0.28	0.02	0.07	0.13
Approach	Top-1 Accuracy	Top-5 Accuracy	Top-10 Accuracy	MRR	MAP	
$CLRQR$	0.81	0.92	0.93	0.86	0.28	
$CLRQR_{WM}$	0.61	0.77	0.78	0.67	0.09	
$CLRQR_{VSM}$	0.32	0.58	0.68	0.42	0.05	

compared with $CLRQR_{VSM}$. From Table 8, we can see that $CLRQR_{WM}$ achieves the second best performance while $CLRQR_{VSM}$ achieves the worst performance.

We apply Wilcoxon signed-rank test (Wilcoxon 1945) to test the significance of improvement of $CLRQR$ over $CLRQR_{WM}$. Table 9 shows that the improvement of $CLRQR$ over the $CLRQR_{WM}$ is significant at the confidence level of 95% score in all evaluated metrics.

$CLRQR$ (i.e., word-embedding based retrieval algorithm) achieves the best performance among the three retrieval algorithms, and thus it is the most suitable algorithm for the task.

RQ2: How effective is our approach $CLRQR$ which uses our domain specific translation method? How much improvement can it achieve over the baseline approach $CLRQR^{BT}$ which uses the basic translation result?

Motivation Accurate translation is a critical step in cross-language information retrieval (Kraaij et al. 2003; Aceves-Pérez et al. 2007). For a given Chinese word, general translation tools return a basic translation result (i.e., the most common translation). Without domain-specific knowledge, the most common translation can be blindly adopted, which may not reflect the developer’s intention. In contrast, our approach takes into account domain-specific knowledge extracted from Stack Overflow discussions to select more-likely domain-specific translation. We would like to confirm the benefits of our domain-specific translation method over the simply use of the most common translation result.

Approach To investigate the effectiveness of our approach $CLRQR$ using our domain specific translation approach, a baseline approach named $CLRQR^{BT}$ is built. $CLRQR^{BT}$ is the same as $CLRQR$, except that it directly uses the basic translation result by the general translation tool as the translation result. As the other settings remain the same, the performance difference reflects the effectiveness of domain-specific translation over the basic translation. We compare the effectiveness of the two translation methods by the Precision@k, Recall@k, Top-k accuracies (k = 1, 5 and 10), MRR, and MAP. To check if the differences in using our domain-specific translation and the basic translation are statistically significant, for each evaluated metric, we apply the Wilcoxon signed-rank test (Wilcoxon 1945) at 95% significance level on the paired metrics by the two translation methods.

Result Among all 120 query Chinese questions, we find that there are six questions for which the translation results generated by our domain-specific translation and the basic translation are the same. For the remaining 114 questions, Table 10 presents the Precision@k, Recall@k, Top-k accuracies (k=1, 5 and 10), MRR and MAP for $CLRQR$ using our domain-specific translation compared with $CLRQR^{BT}$ using the basic translation. We notice that $CLRQR$ using our domain-specific translation achieves the better performance in all evaluated metrics by a substantial margin, compared with $CLRQR^{BT}$ using the basic

Table 9 P-values of $CLRQR$ compared with $CLRQR_{WM}$

	Precision@1	Precision@5	Precision@10	Recall@1	Recall@5	Recall@10
<i>p-value</i>	$2.06 \times e^{-4}$	$2.93 \times e^{-11}$	$6.19 \times e^{-14}$	$5.78 \times e^{-3}$	$2.55 \times e^{-9}$	$2.11 \times e^{-11}$
	Top-1 Accuracy	Top-5 Accuracy	Top-10 Accuracy	MRR	MAP	
<i>p-value</i>	$2.06 \times e^{-4}$	$5.29 \times e^{-4}$	$1.17 \times e^{-3}$	$1.66 \times e^{-11}$	$1.46 \times e^{-4}$	

Table 10 Precision@k, Recall@k, Top-k Accuracies (k=1, 5 and 10), MRR and MAP for *CLRQR* and *CLRQR^{BT}*

Approach	Precision@1	Precision@5	Precision@10	Recall@1	Recall@5	Recall@10
<i>CLRQR</i>	0.81	0.77	0.72	0.05	0.20	0.36
<i>CLRQR^{BT}</i>	0.52	0.38	0.30	0.03	0.10	0.15
Approach	Top-1 Accuracy	Top-5 Accuracy	Top-10 Accuracy	MRR	MAP	
<i>CLRQR</i>	0.81	0.92	0.93	0.86	0.28	
<i>CLRQR^{BT}</i>	0.52	0.72	0.74	0.60	0.07	

translation. Table 11 summarizes the results of the Wilcoxon signed-rank test. The results show that the performance differences are statistically significant for all evaluated metrics.

The proposed domain-specific translation method is effective. It significantly outperforms the simply use of the basic translation result by general translation tool.

RQ3: *How effective is our approach CLRQR in cross-language relevant question retrieval? How much improvement can it achieve over the four baseline approaches?*

Motivation The goal of our approach is to improve the cross-language question retrieval by addressing the key challenges in keyword extraction, domain-specific translation, and query-question relevance ranking in the process of cross-language question retrieval. We want to investigate whether by addressing these challenges our approach can improve the effectiveness of cross-language question retrieval, compared with the baseline approaches which do not explicitly address these challenges.

Approach In this evaluation, we compare the performance of *CLRQR*, the performance of *CLRQR_{vsm}* (i.e., the variant of *CLRQR* that uses vector space model based retrieval algorithm that achieves the worst performance among the three retrieval algorithms), and the performance of the four baseline approaches. The comparison is based on the evaluation metrics Precision@k, Recall@k, Top-k accuracies (k = 1, 5 and 10), MRR, and MAP. To check if the differences in the performance of different approaches are statistically significant, for each evaluated metric, we apply the Wilcoxon signed-rank test (Wilcoxon 1945) at 95% significance level on the paired metrics by our approach *CLRQR* and the best baseline approach.

Result Table 12 presents the Precision@k, Recall@k, Top-k accuracies (k=1, 5 and 10), MRR and MAP for *CLRQR*, *CLRQR_{vsm}* and the four baseline approaches, respectively.

Table 11 P-values of *CLRQR* compared with *CLRQR^{BT}*

	Precision@1	Precision@5	Precision@10	Recall@1	Recall@5	Recall@10
<i>p-value</i>	$7.83 \times e^{-7}$	$1.35 \times e^{-14}$	$3.39 \times e^{-15}$	$2.90 \times e^{-4}$	$1.38 \times e^{-12}$	$2.19 \times e^{-13}$
	Top-1 Accuracy	Top-5 Accuracy	Top-10 Accuracy	MRR	MAP	
<i>p-value</i>	$7.83 \times e^{-7}$	$6.21 \times e^{-6}$	$3.10 \times e^{-5}$	$9.80 \times e^{-15}$	$1.97 \times e^{-7}$	

Table 12 Precision@k, Recall@k, Top-k Accuracies (k=1, 5 and 10), MRR and MAP for *CLRQR* compared with the Other Four Baseline Approaches

Approach	Precision@1	Precision@5	Precision@10	Recall@1	Recall@5	Recall@10
<i>CLRQR</i>	0.81	0.77	0.72	0.05	0.20	0.36
<i>CLRQR_{VSM}</i>	0.32	0.30	0.28	0.02	0.07	0.13
<i>BaselineApproach1</i>	0.21	0.12	0.09	0.01	0.03	0.04
<i>BaselineApproach2</i>	0.00	0.00	0.00	0.01	0.00	0.00
<i>BaselineApproach3</i>	0.43	0.29	0.23	0.02	0.07	0.11
<i>BaselineApproach4</i>	0.27	0.13	0.08	0.01	0.03	0.04
Approach	Top-1 Accuracy	Top-5 Accuracy	Top-10 Accuracy	MRR	MAP	
<i>CLRQR</i>	0.81	0.92	0.93	0.86	0.28	
<i>CLRQR_{VSM}</i>	0.32	0.58	0.68	0.42	0.05	
<i>BaselineApproach1</i>	0.21	0.25	0.25	0.22	0.02	
<i>BaselineApproach2</i>	0.00	0.00	0.00	0.00	0.00	
<i>BaselineApproach3</i>	0.43	0.63	0.67	0.52	0.04	
<i>BaselineApproach4</i>	0.27	0.32	0.34	0.29	0.02	

We notice that *CLRQR_{VSM}* achieves the better performance than the *BaselineApproach1*, *BaselineApproach2* and *BaselineApproach4* in all evaluated metrics. In 8 out of 12 metrics, *CLRQR_{VSM}* achieves slightly better or comparable performance compared with the best baseline approach *BaselineApproach3*.

From Table 12, we notice that the *BaselineApproach3* achieves the best performance among the four baseline approaches, while the *BaselineApproach2* achieves the worst performance (indeed, all the metrics of the *BaselineApproach2* are 0). The main cause for the poor performance of the *BaselineApproach2* is that Stack Overflow search engine is very sensitive to input query. For example, the retrieval result of “difference between A and B” is different from “difference between B and A”. The sensitiveness is also illustrated in Figs. 3 and 4. Thus, retrieving relevant question by Stack Overflow search engine could be very difficult for those developers whose native language is not English due to the use of inappropriate English words and improper query formulation.

Table 12 shows that *BaselineApproach1* outperforms *BaselineApproach2*, and *BaselineApproach3* outperforms *BaselineApproach4*. *BaselineApproach1* and *BaselineApproach3* translate only the title of a given Chinese question, while *BaselineApproach2* and *BaselineApproach4* translate both the title and description of a given Chinese question. The title and description together usually contain more Chinese words. Because general Chinese-English translation tool does not know how to translate Chinese words into domain-specific English words, the more the translated words, the more translation errors may be accumulated, which may affect retrieval accuracy. Furthermore, *BaselineApproach2* and *BaselineApproach4* may recommend questions in the top-10 results for keywords that only appear in the description of a Chinese question. Question description usually contains many technical details which may mislead search engine to return irrelevant questions. This may also degrade the performance of *BaselineApproach2* and *BaselineApproach4*.

We also notice that *BaselineApproach3* outperforms *BaselineApproach1* and *BaselineApproach4* outperforms *BaselineApproach2*. *BaselineApproach1* and

BaselineApproach2 retrieve relevant questions using Stack Overflow search engine. *BaselineApproach3* and *BaselineApproach4* retrieve relevant questions using Google search engine. Our results indicate that Google search engine performs much better and more robustly than Stack Overflow search engine.

The performance of our approach *CLRQR* is much better than the best baseline approach *BaselineApproach3*.

We apply Wilcoxon signed-rank test (Wilcoxon 1945) to compare our approach *CLRQR* with the best baseline approach *BaselineApproach3*. Table 13 shows that the improvement of *CLRQR* over the *BaselineApproach3* is statistically significant at the confidence level of 95% score for all the evaluated metrics.

By addressing the challenges in keyword extraction, domain-specific translation and query-question relevance ranking, our approach *CLRQR* performs much better than the four baseline approaches for cross-language question retrieval.

RQ4: What is the impact of different weights assignments for different kinds of essential information on the performance of question retrieval?

Motivation We divide the essential information of a Chinese question into four kinds: (i) Chinese words in Title (CT), (ii) English words in Title (ET), (iii) Chinese Keywords in Description (CKD), and (iv) English words in Description (ED), and different kinds of query words are given different weights for calculating query-question relevance.

We are interested to investigate the impact of different weight settings on the question retrieval performance.

Approach Our default weight setting for CT, ET, CKD, and ED is 3, 3, 1, and 1, respectively. In this evaluation, we assign CT:ET:CKD:ED with five different weight settings k:k:1:1 (for k=1 to 5) and compare the performance of different settings.

Result Table 14 presents the Precision@k, Recall@k, Top-k accuracies (k=1, 5 and 10), MRR and MAP for the different weight settings. We notice that for most of the evaluation metrics except Recall@1, Top-5 Accuracy and Top-10 Accuracy, the weight setting 3:3:1:1 outperforms the other four weight settings. The weight setting 4:4:1:1 achieves the second best performance. We also apply Wilcoxon signed-rank test to compare our default weight setting 3:3:1:1 with the second best weight setting 4:4:1:1. Table 15 shows that the improvement of the weight setting 3:3:1:1 over the second best weight setting 4:4:1:1. Compared with the weight setting 4:4:1:1, the improvement is statistically significant at the confidence level of 95% confidence level only for Precision@5, Precision@10, Recall@5, Recall@10 and MAP. This indicates that different weight settings can impact the question retrieval performance to certain extent, but often not significantly.

Table 13 P-values of *CLRQR* compared with the Best Baseline Approaches (i.e., *Baseline Approach3*)

	Precision@1	Precision@5	Precision@10	Recall@1	Recall@5	Recall@10
<i>p-value</i>	$2.39 \times e^{-9}$	$2.37 \times e^{-16}$	$2.20 \times e^{-16}$	$3.59 \times e^{-6}$	$7.73 \times e^{-14}$	$6.64 \times e^{-14}$
	Top-1 Accuracy	Top-5 Accuracy	Top-10 Accuracy	MRR	MAP	
<i>p-value</i>	$2.39 \times e^{-9}$	$1.53 \times e^{-7}$	$6.69 \times e^{-7}$	$3.37 \times e^{-16}$	$3.49 \times e^{-9}$	

Table 14 Precision@k, Recall@k, Top-k Accuracies (k=1, 5 and 10), MRR and MAP for the Default Weight Setting 3:3:1:1 Compared with the other Four Weight Settings

Weights	Precision@1	Precision@5	Precision@10	Recall@1	Recall@5	Recall@10
<i>CT:ET:CKD:ED</i>						
1:1:1:1	0.65	0.55	0.39	0.04	0.13	0.20
2:2:1:1	0.76	0.63	0.47	0.04	0.16	0.23
3:3:1:1 (<i>CLRQR</i>)	0.81	0.77	0.72	0.05	0.20	0.36
4:4:1:1	0.78	0.67	0.49	0.05	0.17	0.25
5:5:1:1	0.77	0.66	0.48	0.05	0.17	0.24
Weights	Top-1 Accuracy	Top-5 Accuracy	Top-10 Accuracy	MRR	MAP	
<i>CT:ET:CKD:ED</i>						
1:1:1:1	0.65	0.85	0.87	0.73	0.10	
2:2:1:1	0.76	0.90	0.92	0.81	0.13	
3:3:1:1 (<i>CLRQR</i>)	0.81	0.92	0.93	0.86	0.28	
4:4:1:1	0.78	0.92	0.93	0.84	0.13	
5:5:1:1	0.77	0.91	0.93	0.83	0.13	

The weight setting 1:1:1:1 means that words in question title and words in question description are treated the same and given the same weight. Compared with the other four weight settings where words in question title are given more weight, the performance of the weight setting is much worse. Table 15 also shows that the improvement of the weight setting 3:3:1:1 over the weight setting 1:1:1:1. Compared with the weight setting 1:1:1:1, the improvement is statistically significant at the confidence level of 95% confidence level for all evaluated metrics.

We also use Cohen's d (d) which is an effect size used to indicate the standardized difference between two means (Cohen 1988). In our context, we use Cohen's d (d) to compare the best weight setting 3:3:1:1 to the weight setting 1:1:1:1. Table 16 describes the meaning of different Cohen's d (d) values and their corresponding effectiveness level (Cohen 1988). Table 17 presents the Cohen's d (d) values in terms of all the metrics. Comparing the best weight setting 3:3:1:1 with the weight setting 1:1:1:1, we observe that the Cohen's d (d) values of five metrics are greater than 0.50 and the other five metrics are greater than

Table 15 P-values of the Weight Setting 3:3:1:1 Compared with the Weight Setting 1:1:1:1 and 4:4:1:1

	Precision@1	Precision@5	Precision@10	Recall@1	Recall@5	Recall@10
1:1:1:1 <i>p-value</i>	$1.053 \times e^{-4}$	$1.193 \times e^{-8}$	$5.980 \times e^{-13}$	$5.439 \times e^{-4}$	$8.312 \times e^{-9}$	$2.45 \times e^{-12}$
4:4:1:1 <i>p-value</i>	0.172	$1.280 \times e^{-5}$	$1.340 \times e^{-12}$	0.186	$4.900 \times e^{-5}$	$9.560 \times e^{-12}$
	Top-1 Accuracy	Top-5 Accuracy	Top-10 Accuracy	MRR	MAP	
1:1:1:1 <i>p-value</i>	$1.053 \times e^{-4}$	$9.828 \times e^{-3}$	$1.844 \times e^{-2}$	$5.046 \times e^{-4}$	$9.556 \times e^{-12}$	
4:4:1:1 <i>p-value</i>	0.172	1.000	1.000	0.0907	$1.230 \times e^{-10}$	

Table 16 Cohen's d and the effectiveness level

Cohen's d ($ d $)	Effectiveness level
$ d < 0.20$	Small
$0.20 \leq d < 0.50$	Medium
$0.50 \geq d $	Large

0.20. This indicates that the improvement of the best weight setting 3:3:1:1 over the weight setting 1:1:1:1 is substantial. This result actually confirmed our observation that the question title usually sums up the core issue of the question better than the question description. Therefore, the weights of the words in the title should be set higher than that of the words in the description.

Assigning a high weight to the words in title leads to a better result. The best performing configuration is 3:3:1:1, and its results are significantly better than 1:1:1:1, while only being significantly better than 4:4:1:1 for some of the metrics considered.

RQ5: What is the time efficiency of our CLRQR system?

Motivation The runtime efficiency of the proposed approach will affect its practical usage. In our approach, the main time cost is to build domain-specific vocabulary and to retrieve relevant questions on the fly. The time for building domain-specific vocabulary refers to the time required to count the term frequency of each word in our dataset. Question retrieval time refers to the time to extract Chinese and English words from the input question, translate Chinese keywords on-the-fly, compute keyword weights, formulate English query, and finally retrieve relevant questions in question repository.

Approach The experimental environment is an Intel(R) Core(TM) i7 2.5 GHz laptop with 16GB RAM running OS X Version 10.11.1 (64-bit). To answer RQ4, we record the start time and the end time of the program to build domain-specific vocabulary, and the start time and the end time to retrieve questions for each query.

Result For building domain-specific vocabulary, it takes about 8 seconds to analyze 30,000 questions on average. For question retrieval time, on average our approach needs about 2 seconds to return relevant questions in the repository of 100,000 questions. Since, the relevance calculation between a given query and each English question in the repository is independent with one another, it is possible to reduce question retrieval time by distributed computing.

Our approach is efficient. The domain-specific vocabulary building time can be completed in about 8 seconds. On average, question retrieval can return relevant questions in

Table 17 Cohen's d of the Weight Setting 3:3:1:1 Compared with the Weight Setting 1:1:1:1

	Precision@1	Precision@5	Precision@10	Recall@1	Recall@5	Recall@10
1:1:1:1 <i>Cohen's d</i>	0.446	0.780	1.240	0.352	0.984	1.730
	Top-1 Accuracy	Top-5 Accuracy	Top-10 Accuracy	MRR	MAP	
1:1:1:1 <i>Cohen's d</i>	0.446	0.286	0.247	0.457	1.560	



Fig. 8 A Chinese Question on V2EX

the repository of 100,000 questions in about 2 seconds, which still have room for future improvement.

6 Discussion

This section first discusses about the generality of our approach whether our approach can be applied for some other domains. Then, we presents the qualitative analysis of some representative examples in our experiments and discusses threats to validity of our experiment.

6.1 Qualitative Analysis

Our qualitative analysis of representative examples aims to answer the following three questions:

Q1: Why does $CLRQR$ perform better than $CLRQR_{WM}$? Why does $CLRQR_{WM}$ perform better than $CLRQR_{VSM}$?

Figure 8 presents a Chinese question from V2EX¹¹ (a Chinese Q&A website for computer programming). The meaning of this Chinese question is to ask for the advantage of Play framework. The translation of the Chinese word “优势” in the question is “advantage”. The most relevant English question from Stack Overflow returned by $CLRQR$ is shown in Fig. 9. The Stack Overflow question asks about the benefit from getter and setter in Play framework. We observe that word embedding captures the semantic similarity between the word “advantage” and “benefit” very well. As such, the cosine similarity between the word embedding vectors of “advantage” and “benefit” is very high. In contrast, $CLRQR_{WM}$ and $CLRQR_{VSM}$ cannot overcome the lexical gap between the query and the question.

For the reason why $CLRQR_{WM}$ performs better than $CLRQR_{VSM}$, we use an example to illustrate it. Assume the $Vector_{Query}$ equals to $[0.2, 0.2, 0.2, 0.2]$ which means there are four query words and they have 20% contribution for semantic expression respectively.

- **Illustrative example:** Assume $Vector_{Q1}$ equals to $[0.15, 0.15, 0.15, 0.15]$ and $Vector_{Q2}$ equals to $[0.1, 0.1, 0.1, 0.1]$. The cosine similarity between $Vector_{Query}$ and $Vector_{Q1}$ is 1. The cosine similarity between $Vector_{Query}$ and $Vector_{Q2}$ is also

¹¹ A Chinese question on V2EX, available at <https://www.v2ex.com/t/47663>

Benefit from generated getters and setters in Play! framework

▲ The Play! framework generates getters and setters for each public field of a model class at runtime.

9



1

```
public class Product {
    public String name;
    public Integer price;
}
```

will be transformed to

Fig. 9 An Relevant English Question on *Stack Overflow* question ID=7576550

1. Thus, $CLRQR_{VSM}$ considers that $Vector_{Q_1}$ and $Vector_{Q_2}$ have the same relevance for $Vector_{Query}$. Obviously, $Vector_{Q_1}$ is more relevant with $Vector_{Query}$, because each query word in $Vector_{Q_1}$ has higher weight than the corresponding word in $Vector_{Q_2}$.

Cosine similarity only measures the direction similarity between the vectors. However, in the above example, $Vector_{Q_1}$ is closer to $Vector_{Query}$ in terms of Euclidean distance, compared with $Vector_{Q_2}$. As such, $CLRQR_{WM}$ which computes a variant of Euclidean distance between the query and the question can better reflect the relevance between the query and the question than $CLRQR_{VSM}$. Our analysis suggests that cosine similarity is not suitable for the question retrieval task, compared with other distance metrics.

Q2: Why does our domain specific translation method perform better than the basic translation $CLRQR^{BT}$?

Among 80 Chinese questions, there are only four questions whose translation results are the same by our domain-specific translation method and the general translation $CLRQR^{BT}$. Figure 10 presents a Chinese question from Segment Fault.¹² This Chinese question asks about how to handle stream closed exception. The basic translation result of Chinese word “异常” is “abnormal” which is not the proper translation result in software engineering context. Our proposed approach $CLRQR^{Filter}$ can obtain the correct domain-specific translation “exception” for the Chinese word. Our analysis confirms that it is usually difficult to obtain the right translation result without considering domain-specific knowledge.

Q3: Why does our approach perform better than the four baseline approaches?

Figure 11 presents a Chinese question from V2EX.¹³ This Chinese question asks about the route error of Play framework. Our approach extracts essential information from the Chinese question and then translates Chinese keywords to simulate the English query that users may enter as follows (sorted by their weights in descending order):

play, framework, route, error, api, spring

¹²A Chinese question on *SegmentFault* available at <https://SegmentFault.com/q/1010000003408795>

¹³A Chinese question on *V2EX*, available at <https://www.v2ex.com/t/137913>

segmentfault 头条 问答 专栏 职位 活动

home feed javascript php python java mysql ios android node.js html5 linux c++ css3 git

问 Stream closed异常

java **Magiche** 2015年08月21日提问

我有两个方法，分别解析两段接收到的xml，我确定两个方法的参数request是不一样的。方法1可正确执行，方法2报错Stream closed。我把inputStream.close()和inputStream = null注释掉还是一样。方法如下，

方法1:

```
public static Map<String, String> parseXml(HttpServletRequest request) throws Exception {
    // 将解析结果存储在HashMap中
    Map<String, String> map = new HashMap<String, String>();
    // 从request中取得输入流
    InputStream inputStream = request.getInputStream();
    // 读取输入流
    SAXReader reader = new SAXReader();
    Document document = reader.read(inputStream);
    ...
    // 释放资源
    inputStream.close();
    inputStream = null;
    return map;
}
```

Fig. 10 A Chinese Question on *SegmentFault*

The 10 questions returned by our approach are all actually relevant. The result shows that the extracted keywords well represent the core issue of the given question. We observe that our approach can capture Chinese and English keywords well at the same time if the Chinese question asks state their question clearly and sum up the core issue in the title. Moreover, we find that the sentences in question description are often incomplete, and thus it does not have much impact on our keyword extraction. On the other hand, if we translate the title of this Chinese question into English by Youdao translation API and then retrieve relevant questions in Stack Overflow by Google search engine, it returns only one relevant question.

V2EX

V2EX > Java

碰到 play framework route 错误，大家有没有遇到过

woshifyz · 2014-10-09 23:20:16 +08:00 · 1078 次点击

这是一个创建于 466 天前的主题，其中的信息可能已经有所发展或是发生改变。

用的是play framework + spring，现在遇到一个路由的问题，大概是这个样子的：

有A、B两个api，同时大量的请求这两个api，在log中会显示在B的处理方法中接收到了本来该发向A的请求，不知道大家有没有碰到过

Fig. 11 A Chinese Question on *V2EX*

6.2 Threats to Validity

There are several threats that may potentially affect the validity of our study. Threats to internal validity relate to errors in our experiments and implementation. We have double checked our experiments and implementation. We have also manually checked the retrieved questions in our dataset to ensure that they are really tagged with *java*. Still, there could be errors that we have not noticed.

Threats to external validity relates to the generalizability of our results. We conducted a user study to evaluate whether the retrieved relevant questions are actually relevant or not. To reduce this threat, we invite 5 master students for the user study. All participants have industrial experience in Java programming (ranging from 3–6 years) and pass the national College-English-Test Level 4. In the future, we plan to reduce this threat further by analyzing more Chinese questions and building larger English question repository.

7 Related Work

In this paper, we propose a new approach to retrieve cross-language relevant question in software engineering(SE). Thus, we first describe cross-language studies and then give a review on information retrieval in SE. In order to improve the results returned by an search algorithm, we also propose a new query formulation strategy by building a weighted query word list for a given technique question. Therefore, we also review some previous studies on query formulation for software artifacts.

7.1 Cross-Language Studies in Software Engineering

Many studies have been carried out on cross-language issues in SE. Xia et al. propose a cross-language bug localization algorithm named *CrosLocator* (Xia et al. 2014). *CrosLocator* uses multiple translators to convert a non-English textual description of a bug report into English - each bug report would then have multiple translated versions. For each translated version, *CrosLocator* applies a bug localization technique to rank source code files. Finally, *CrosLocator* combines the multiple ranked lists of source code files. Hayes et al. propose a translation-based method for traceability recovery (Hayes et al. 2011). They use Google translator to translate Italian words into English and then recover the links. Chang and Lee present a cross-language video Q&A system i.e., CLVQ, which could process the English questions, and find answers in Chinese videos (Shepherd et al. 2005). Saggion et al. focus on the resources that are made available for the research community (Saggion et al. 2002). They provide data and tools for evaluation of extractive, non-extractive, single and multi-document summarization.

Our work also focuses on cross-language question retrieval between Chinese and English and aims improve the accuracy of translation of domain-specific Chinese words based on domain knowledge derived from Stack Overflow.

7.2 Cross-Language Information Retrieval

The defining characteristic of Cross-language Information Retrieval (CLIR) is that queries and documents are expressed in different languages (Jones et al. 1999). Much research work focuses on improving the translation performance between the two different languages. CLIR techniques can be classified into different categories based on different translation

approaches: (i) Dictionary-based techniques, (ii) Parallel corpora based techniques, (iii) Comparable corpora based techniques and (iv) Machine translator based techniques (Thai 2007).

Among these four kinds of translation approaches, dictionary-based technique is the most relevant to our work as we formulate a query as a scored word list. Dictionary-based techniques utilize machine readable dictionaries (MRD), bilingual word lists, or other lexicon resources to translate the query terms by replacing them with their target language equivalents. Hull and Grefenstette compare the effectiveness of a monolingual IR system, and CLIR systems that translate the queries using either an automatically constructed bilingual MRD, or a manually constructed dictionary (Hull and Grefenstette 1996). The study shows that the recognition and translation of multi-word expressions and phrases are crucial to success in CLIR. Another observation is that lexicon ambiguity is a great cause for translation errors. A word can often be translated into several meanings; not all were intended in the query. A CLIR system will have either use all the translations or be able to choose among the options and find the ones that best represent the original query. Kluck and Gey (Kluck and Gey 2001b) point out that in many cases there exists a clear difference between the domain-specific meaning and the common meaning of a word. This means that it can be difficult to use the common translation result of a word for domain-specific information retrieval. Thus, the drawback of dictionary-based techniques will be magnified in domain-specific translation tasks. In the paper, we use popular translation tools (i.e., Youdao Translate and Google Translate) as our “dictionary” to improve the translation results. Furthermore, we extract domain knowledge by building a domain-specific vocabulary to overcome lexicon ambiguity.

7.3 Information Retrieval in Software Engineering

Many studies have been reported on information retrieval in SE (e.g., Canfora and Cerulo (2005), Lucia et al. (2007), Marcus et al. (2004), Poshyvanyk et al. (2007), Čubranić and Murphy (2003), Xu et al. (2017a), Yang et al. (2016), Zhang et al. (2015, 2016, 2017). Marcus et al. propose an information retrieval approach to concept Location in source code (Marcus et al. 2004). Poshyvanyk et al. recast the problem of feature location in source code as a decision-making problem in the presence of uncertainty. They point out that the solution to feature location can be formulated as a combination of the opinions of different experts (Poshyvanyk et al. 2007). Canfora and Cerulo outline an approach to in automated bug assignment based on information retrieval in which they report recall levels of around 20% for Mozilla (Canfora and Cerulo 2005).

Čubranić and Murphy apply information retrieval as well as other matching techniques to recover the implicit traceability among different kinds of artifacts of open source projects (i.e., source file revisions, change or bug tracks, communication messages, and documents) (Čubranić and Murphy 2003). They develop the Hipikat tool which can recommend some software development artifacts to newcomers under current tasks.

Lucia et al. observe that the main drawback of existing software artifact management systems is the lack of automatic or semi-automatic traceability link generation and maintenance (Lucia et al. 2007). Hence, they improve an artifact management system with a traceability recovery tool based on Latent Semantic Indexing (LSI), an information retrieval technique. Even more, they assess the strengths and limitations of LSI for traceability recovery and devise the need for an incremental approach. Kluck and Gey describe the domain-specific cross-language information retrieval (CLIR) task of CLEF, why and how it is important and how it differs from general cross-language retrieval problem associated with the general CLEF collections (Kluck and Gey 2001a).

Our work mainly focuses on how to adapt information retrieval technology with the heuristic for extracting information from different parts of a questions for better retrieving and ranking relevant questions.

7.4 Query Formulation Strategies for Searching Software Artifacts

Query formulation is an essential part of successful information retrieval. Various approaches have been proposed, which fall in two main categories: query reduction and query expansion. There are many software engineering tasks that can be addressed by text retrieval techniques, such as traceability link recovery, feature location, refactoring, code reuse, etc. Haiduc et al. (2013a). A common issue with all tasks is that the quality of retrieval results depend largely on the quality of the query. When a query is vague or of poor quality, it has to be reformulated. Query reformulation can be a difficult task for someone who has difficulties in writing a good query in the first place. Haiduc et al. develop an Eclipse plugin which is able to automatically detect the quality of a text retrieval query and to propose reformulations for it, when needed, in order to improve the results of TR-based code search (Haiduc et al. 2013b). Marcus et al. use Latent Semantic Indexing in order to determine the most similar terms to the query from the source code and include these similar terms in the query (Marcus et al. 2004). Yang and Tan use the context in which query words are found in the source code to extract synonyms, antonyms, abbreviations and related words for query reformulation (Yang and Tan 2012). Hill et al. also use word context in order to extract possible query expansion terms from the code (Hill et al. 2009). Shepherd et al. build a code search tool that expands search queries with alternative words learned from verb-direct object pairs (Shepherd et al. 2007).

We find that it is difficult for Chinese developers to formulate appropriate English queries for search. Thus, we propose a query reformulation strategy to formulate an English query for a given Chinese query. Considering different contribution of a word to express the core issue a question, we assign different weights to each kind of words depending on their location in the question and their languages. Thus, the problem we aim to tackle and our query reformulation strategy are different from existing work.

8 Conclusion and Future Work

We propose a novel approach to retrieve relevant English questions for a given Chinese question.

We mine domain-specific knowledge from Stack Overflow to improve the accuracy of translation of domain-specific Chinese words. Considering the difference between question title and description, we assign query words from title and descriptions with different weights in query formulation and query retrieval. To overcome the lexical gap issue, we propose to adopt neural language model (word embeddings). All the steps of our approach are automated, and thus it can help developers save time in terms of query translation, query formulation, and question retrieval. As a result, it can potentially help Chinese developers improve their efficiency to solve the technical problems. Our experiments involve questions of three programming languages, i.e., Java, Python and .NET. Our results demonstrate not only the effectiveness of our approach but also the generality of our approach for questions related to different programming technologies.

In our experiments, we see some retrieved questions are not very relevant to the query due to the limitation of the question repository we build for experimentation. We will crawl

more questions on Stack Overflow or some other Q&A sites. This will help improve the relevance and usefulness of retrieved questions. Furthermore, we observe that most of the questions in Stack Overflow contain code. In the current approach, we treat code as natural language text. In the future, we can extract code segments of questions and process them in a different way from regular English texts.

Another improvement is to expand our domain-specific stop words list and vocabulary to improve the accuracy of domain-specific translation.

Acknowledgment This work was partially supported by NSFC Program (No. 61602403 and 61572426).

References

- Aceves-Pérez RM, Montes-y Gómez M, Villaseñor-Pineda L (2007) Enhancing cross-language question answering by combining multiple question translations. In: *Computational Linguistics and Intelligent Text Processing*, Springer, pp 485–493
- Baeza-Yates R, Ribeiro-Neto B et al (1999) *Modern information retrieval*, vol 463. ACM Press, New York
- Bao L, Lo D, Xia X, Li S (2017) Automated android application permission recommendation. *Sci China Inf Sci* 60(9):092,110
- Canfora G, Cerulo L (2005) How software repositories can help in resolving a new change request. *STEP* 2005:99
- Cohen J (1988) *Statistical power analysis for the behavioral sciences*. hillsdale. Lawrence Earlbaum Associates, New Jersey, p 2
- Cui H, Wen JR, Nie JY, Ma WY (2002) Probabilistic query expansion using query logs. In: *Proceedings of the 11th international conference on World Wide Web*, ACM, pp 325–332
- Haiduc S, Bavota G, Marcus A, Oliveto R, De Lucia A, Menzies T (2013a) Automatic query reformulations for text retrieval in software engineering. In: *2013 35th international conference on software engineering (ICSE)*, IEEE, pp 842–851
- Haiduc S, De Rosa G, Bavota G, Oliveto R, De Lucia A, Marcus A (2013b) Query quality prediction and reformulation for source code search: The refoqus tool. In: *Proceedings of the 2013 international conference on software engineering*, IEEE Press, pp 1307–1310
- Harkness (2017) Why are some chinese students who have learnt english for years still poor in english? <https://goo.gl/7ltMLy>
- Harris ZS (1954) Distributional structure. *Word* 10(2-3):146–162
- Hayes JH, Sultanov H, Kong WK, Li W (2011) Software verification and validation research laboratory (svvrl) of the university of kentucky: traceability challenge 2011: language translation. *Selabnet-labukyedu* pp 50–53
- Hiemstra D, De Jong F, Kraaij W (1997) A domain specific lexicon acquisition tool for cross-language information retrieval. In: *Computer-Assisted Information Searching on Internet*, LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, pp 255–268
- Hill E, Pollock L, Vijay-Shanker K (2009) Automatically capturing source code context of nl-queries for software maintenance and reuse. In: *IEEE 31st international conference on software engineering*, 2009. ICSE 2009. IEEE, pp 232–242
- Hull DA, Grefenstette G (1996) A dictionary-based approach to multilingual informaion retrieval. In: *Proceedings of the 19th international conference on research and development in information retrieval*, pp 49–57
- Jones G, Sakai T, Collier N, Kumano A, Sumita K (1999) A comparison of query translation methods for english-japanese cross-language information retrieval. In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, pp 269–270
- Jui SL (2010) *Innovation in China: the Chinese software industry*. Routledge, Abingdon
- Kluck M, Gey FC (2001a) The domain-specific task of clef - specific evaluation strategies in cross-language information retrieval. In: *Peters C. (ed) Proceedings of the CLEF 2000 evaluation forum*, pp 48–56
- Kluck M, Gey FC (2001b) The domain-specific task of clef-specific evaluation strategies in cross-language information retrieval. In: *Cross-Language Information Retrieval and Evaluation*, Springer, pp 48–56

- Kraaij W, Nie JY, Simard M (2003) Embedding web-based statistical translation models in cross-language information retrieval. *Comput Linguist* 29(3):381–419
- Liu X, Gong Y, Xu W, Zhu S (2002) Document clustering with cluster refinement and model selection capabilities. In: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, pp 191–198
- Lucia AD, Fasano F, Oliveto R, Tortora G (2007) Recovering traceability links in software artifact management systems using information retrieval methods. *ACM Trans Softw Eng Methodol* 16(4):50. *Acm Transactions on Software Engineering & Methodology* 16
- Maaten LVD, Hinton G (2008) Visualizing data using t-sne. *J Mach Learn Res* 9(Nov):2579–2605
- Marcus A, Sergeev A, Rajlich V, Maletic JI (2004) An information retrieval approach to concept location in source code. In: 11th working conference on reverse engineering, 2004. Proceedings. IEEE, pp 214–223
- Mihalcea R, Tarau P (2004) Textrank: Bringing order into texts. *Association for Computational Linguistics*
- Mihalcea R, Corley C, Strapparava C (2006) Corpus-based and knowledge-based measures of text semantic similarity. In: National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16–20, 2006, Boston, Massachusetts, USA, pp 775–780
- Mikolov T, Chen K, Corrado G, Dean J (2013a) Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781
- Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013b) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119
- Peñas A, Magnini B, Forner P, Sutcliffe R, Rodrigo Á, Giampiccolo D (2012) Question answering at the cross-language evaluation forum 2003–2010. *Lang Resour Eval* 46(2):177–217
- Porter MF (1980) An algorithm for suffix stripping. *Program* 14(3):130–137
- Poshyvanyk D, Gueheneuc YG, Marcus A, Antoniol G, Rajlich VC (2007) Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval. *IEEE Trans Softw Eng* 33(6):420–432
- Řehůřek R, Sojka P (2010) Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, ELRA, Valletta, Malta, pp 45–50. <http://is.muni.cz/publication/884893/en>
- Resnik P, Melamed ID (1997) Semi-automatic acquisition of domain-specific translation lexicons. In: Proceedings of the fifth conference on Applied natural language processing, Association for Computational Linguistics, pp 340–347
- Saggion H, Radev D, Teufel S, Lam W, Strassel SM (2002) Developing infrastructure for the evaluation of single and multi-document summarization systems in a cross-lingual environment. *Ann Arbor* 1001(48):109–1092
- Salton G, Buckley C (1988) Term-weighting approaches in automatic text retrieval. *Inf Process Manag* 24(5):513–523
- Shepherd D, Pollock L, Tourwé T (2005) Using language clues to discover crosscutting concerns. *Acm Sigsoft Soft Engineer Notes* 30:1–6
- Shepherd D, Fry ZP, Hill E, Pollock L, Vijay-Shanker K (2007) Using natural language program analysis to locate and understand action-oriented concerns. In: Proceedings of the 6th international conference on Aspect-oriented software development, ACM, pp 212–224
- Tan PN et al (2006) Introduction to data mining. Pearson Education, London
- Thai P (2007) An introduction to cross-language information retrieval approaches. Web.simmons.edu
- Čubranić D, Murphy GC (2003) Hipikat: recommending pertinent software development artifacts. In: 25th international conference on software engineering, 2003. Proceedings. pp 408–418
- Wilcoxon F (1945) Individual comparisons by ranking methods. *Biom Bull* 1(6):80–83. JSTOR
- Xia X, Lo D (2017) An effective change recommendation approach for supplementary bug fixes. *Autom Softw Eng* 24(2):455–498. Springer
- Xia X, Lo D, Wang X, Zhang C, Wang X (2014) Cross-language bug localization. In: Proceedings of the 22nd International Conference on Program Comprehension, ACM, pp 275–278
- Xia X, Lo D, Wang X, Yang X (2015) Who should review this change?: Putting text and file location analyses together for more accurate recommendations. In: 2015 IEEE international conference on software maintenance and evolution (ICSME), IEEE, pp 261–270
- Xu B, Xing Z, Xia X, Lo D, Wang Q, Li S (2016) Domain-specific cross-language relevant question retrieval. In: Proceedings of the 13th International Workshop on Mining Software Repositories, ACM, pp 413–424

- Xu B, Xing Z, Xia X, Lo D (2017a) Answerbot - automated generation of answer summary to developers technical questions. In: Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, IEEE, p Accepted
- Xu B, Xing Z, Xia X, Lo D, Le XBD (2017b) Xsearch: a domain-specific cross-language relevant question retrieval tool. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ACM, pp 1009–1013
- Yang J, Tan L (2012) Inferring semantically related words from software context. In: Proceedings of the 9th IEEE Working Conference on Mining Software Repositories, IEEE Press, pp 161–170
- Yang X, Lo D, Xia X, Bao L, Sun J (2016) Combining word embedding with information retrieval to recommend similar bug reports. In: 2016 IEEE 27th international symposium on software reliability engineering (ISSRE), IEEE, pp 127–137
- Zhang Y, Lo D, Xia X, Sun JL (2015) Multi-factor duplicate question detection in stack overflow. *J Comput Sci Technol* 30(5):981–997
- Zhang Y, Lo D, Xia X, Le TDB, Scanniello G, Sun J (2016) Inferring links between concerns and methods with multi-abstraction vector space model. In: 2016 IEEE international conference on software maintenance and evolution (ICSME), IEEE, pp 110–121
- Zhang Y, Lo D, Kochhar PS, Xia X, Li Q, Sun J (2017) Detecting similar repositories on github. In: 2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER), IEEE, pp 13–23
- Zhou J, Zhang H, Lo D (2012) Where should the bugs be fixed?-more accurate information retrieval-based bug localization based on bug reports. In: Proceedings of the 34th International Conference on Software Engineering, IEEE Press, pp 14–24



Bowen Xu is a PhD student at School of Information Systems, Singapore Management University advised by Associate Professor David Lo. He received his M.Eng. in College of Software Technology, Zhejiang University in 2017. His research interests are in software engineering area. He has been working on cross-language information retrieval techniques, empirical studies and text mining in software engineering.



Zhenchang Xing is the senior lecturer at the research school of computer science, Australian National University, Australia. Dr. Xing's research interests include software engineering and human-computer interaction. His work combines software analytics, behavioral research methods, data mining techniques, and interaction design to understand how developers work, and then build recommendation or exploratory search systems for the timely or serendipitous discovery of the needed information.



Xin Xia will join the Faculty of Information Technology, Monash University, Australia as a lecturer (equivalent to U.S. assistant professor) from January, 2018. Prior to joining Monash University, he was a post-doctoral research fellow in the software practices lab at the University of British Columbia in Canada, and a research assistant professor at Zhejiang University in China. Xin received both of his Ph.D and bachelor degrees in computer science and software engineering from Zhejiang University in 2014 and 2009, respectively. To help developers and testers improve their productivity, his current research focuses on mining and analyzing rich data in software repositories to uncover interesting and actionable information.



David Lo received his PhD degree from the School of Computing, National University of Singapore in 2008. He is currently an Associate Professor in the School of Information Systems, Singapore Management University. He has close to 10 years of experience in software engineering and data mining research and has more than 200 publications in these areas. He received the Lee Foundation Fellow for Research Excellence from the Singapore Management University in 2009, and a number of international research awards including several ACM distinguished paper awards for his work on software analytics. He has served as general and program co-chair of several prestigious international conferences (e.g., IEEE/ACM International Conference on Automated Software Engineering), and editorial board member of a number of high-quality journals (e.g., Empirical Software Engineering).



Shanping Li received his Ph.D. degree from the College of Computer Science and Technology, Zhejiang University in 1993. He is currently a professor in the College of Computer Science and Technology, Zhejiang University. His research interests include Software Engineering, Distributed Computing, and the Linux Operating System.