Question Answering for the Operation of Software Applications: A Document Retrieval Approach

Atsushi FUJII†a), Member and Seiji TAKEGATA††, Nonmember

SUMMARY Reflecting the rapid growth of information technology, the configuration of software applications such as word processors and spreadsheets is both sophisticated and complicated. It is often difficult for users to identify relevant functions in the online manual for a target application. In this paper, we propose a method for question answering that finds functions related to the user’s request. To enhance our method, we address two “mismatch” problems. The first problem is associated with a mismatch in vocabulary, where the same concept is represented by different words in the manual and in the user’s question. The second problem is associated with a mismatch in function. Although the user may have a hypothetical function for a purpose in mind, this purpose can sometimes be accomplished by other functions. To resolve these mismatch problems, we extract terms related to software functions from the Web, so that the user’s question can be matched to the relevant function with high accuracy. We demonstrate the effectiveness of our method experimentally.

key words: question answering, software applications, information retrieval, mismatch problems, expansion methods

1. Introduction

Reflecting the rapid growth of information technology, the configuration of software applications such as word processors and spreadsheets is both sophisticated and complicated. It is often difficult for users to identify relevant functions in the online manual for a target application. In this paper, we propose a method for question answering (QA) that finds functions related to the user’s request. To enhance our method, we address two “mismatch” problems.

The first problem is associated with a “vocabulary mismatch”, where the same concept is represented by different words in the manual and in the user’s question. For example, the user of a word processor may use the term “margin” to refer to the region referred to as the “header” in the manual.

The second problem is associated with a “function mismatch”. Although the user may have a hypothetical function for a purpose in mind, this purpose can sometimes be accomplished by other functions. For example, in a word-processing software package such as Microsoft Word, character spacing is automatically determined via the word wrap function. A user wishing to adjust the character spacing may search for a function related to the size of the gap between characters, whereas the problem can actually be resolved by adjusting the format of the paragraph in question.

To resolve these mismatch problems, we propose a Web-based expansion method. We extract terms related to software functions from the Web, so that questions can be matched to any relevant functions with high accuracy. We demonstrate the effectiveness of our method experimentally.

This paper is based on our previous study [1] and additional experimental results.

Section 2 discusses QA-related research. Section 3 explains the operation of our method and Sect. 4 describes the evaluation of its effectiveness.

2. Related Work

QA has been explored by the natural language processing (NLP), information retrieval (IR), and artificial intelligence (AI) communities. There are two QA systems related to the operation of software. The Berkeley UNIX Consultant project [2] is an AI-oriented system that uses a knowledge base to answer questions about UNIX commands. The Dialog Navigator [3] is an NLP/IR-oriented system that searches a collection of Microsoft support documents for documents that match a particular question, using a number of dictionaries that relate questions and documents. However, both these systems require manual compilation of the knowledge bases and dictionaries, which is expensive. In contrast, our system automatically extracts related terms from the Web to address the two mismatch problems.

Although a number of open-domain QA methods have been proposed in the Text REtrieval Conference (TREC), their main purpose is to answer factoids and supply definitions [4], [5]. Our purpose is to deal with “how-to” questions. Although a number of open-domain QA methods to answer how-to questions such as “How is a site nominated and selected as a World Heritage?” have been proposed in NTCIR [6], these methods cannot find answers that are not in a document collection. In contrast, our domain-specific method, which searches for answers from among a fixed number of candidates, can usually find answers to questions with respect to a target application.

The idea of expanding target documents has been studied for spoken document retrieval (SDR) and conventional text-to-text ad hoc retrieval. Because the purpose of SDR is to search speech archives by text queries, automatic speech recognition is used to transcribe the target speech archives.
and the transcriptions are indexed as surrogate documents. However, because of speech recognition errors, the vocabulary mismatch problem is crucial in SDR. To resolve this problem, Singhal and Pereira [7] searched an external corpus for documents related to transcriptions for expansion purposes, and showed the retrieval effectiveness using the TREC-7 SDR test collection [8]. In practice, because the target speech archives comprise broadcast news, newspaper articles published during the same period as this broadcast news were used as a corpus. Although our expansion method is similar to the above method from a technical point of view, our research is different from Singhal and Pereira [7] in terms of the following perspectives.

First, Singhal and Pereira [7] addressed the vocabulary mismatch problem caused by speech recognition errors, whereas we address the vocabulary mismatch problem caused by the user’s knowledge and also address the function mismatch problem in QA. Second, the length of target documents is substantially different. The average length of target documents in the TREC-7 SDR was 269 words [8], whereas the description of a function for a software application is usually much shorter. For example, the average number of words in function descriptions for Word, Excel, and PowerPoint (PPT) in Microsoft Office 2000 is 12.2 (we will describe details of function descriptions for these applications in Sect. 3.2). It is potentially difficult to search for related documents for a short target document with high accuracy. However, we will show that our expansion method is effective even when target documents are short. Third, the experimental result in Singhal and Pereira [7] was obtained using only 19 queries, but we will use 353 questions to draw more reliable conclusions (see Sect. 4.1 for details). Finally, Singhal and Pereira [7] showed that their document expansion method is not effective if a corpus that includes documents related to the target documents is not available, whereas we use the Web as a corpus to alleviate this problem.

For text-to-text retrieval, Billerbeck and Zobel [9] and Tao et al. [10] independently explored document expansion methods that do not use an external corpus and expand each document using other documents in the same collection. Billerbeck and Zobel [9] experimentally showed that their document expansion method was comparable with a method with no expansion and was not as effective in retrieval as a query expansion method. Tao et al. [10] expanded target documents to estimate parameter values accurately in the language model retrieval framework.

However, our research is different from both Billerbeck and Zobel [9] and Tao et al. [10] in terms of the first and second differences discussed above for Singhal and Pereira [7]. First, unlike previous studies, we address both the vocabulary and function mismatch problems in QA. Second, the length of target documents in our research is different from those in previous studies. Billerbeck and Zobel [9] and Tao et al. [10] used the TREC ad hoc retrieval data set, where the average number of words per document is approximately 300 [11]. Tao et al. [10] purposefully reduced the length of each target document and showed that their document expansion method was effective even for short documents. For this experiment, they used the Associated Press (AP) newswires in the period 1998–1990, each of which consisted of an average of 445 words [11]. However, because each AP document was reduced to 30% of its original length, the average number of words in these artificially produced short documents was 133, which is still 10 times longer than the average length of function descriptions for Word, Excel, and PPT in Microsoft Office 2000.

3. A QA Method for Software Applications

3.1 Overview

Figure 1 depicts the overall design of our QA method, in which the left-hand and right-hand regions correspond to the offline and online processes, respectively. Although our method is currently implemented in Japanese, our methodology is language independent. We explain our method below, in terms of Fig. 1.

Given a list of all of the functions for a target software application such as a word processor or spreadsheet, the offline process in our method produces indexes for the descriptions of the functions. Example descriptions are “open the file” and “print the document”, which are usually provided as menu items in the software application.

Given a question about the target application, the online process in our method generates a ranked list of functions that are related to the question. In other words, we regard the description of each function as being a single document, and we recast the QA process as a document-retrieval process. For example, functions for the manipulation of fonts will be retrieved by the question “How can I change the font size?”.

However, because the description of each function involves a small number of words, any mismatch in vocabulary between the description of each function and the question is crucial. As discussed in Sect. 1, the function mis-
match problem is also crucial.

To increase the possibility that a question matches the relevant function, we expand the description of each function by using related terms. First, we use the description of each function as a query to search the Web for related pages. It is expected that the retrieved pages will include any expository text for the function in question. It is also expected that these pages will include nontechnical words that novice users may use. Then we extract the relevant fragments of the pages (i.e., passages), aiming to discard irrelevant information. We assume that the extracted passages will include related terms for the function. Finally, we index the description of each function together with the corresponding passages.

Using this method, recall improves but precision is potentially sacrificed. Therefore, we index both the original and the expanded descriptions of the functions to produce two indexes (labeled “index1” and “index2” in Fig. 1). Given a question, we use these indexes to produce independently two ranked lists of functions. Then we integrate these lists and present a final ranked list to the user. In addition, because each function is associated with its related passages, the user can read these passages to understand better the details of each function. This feature is particularly important when a user must identify relevant functions in the ranked list.

Currently, our method targets questions that can be resolved using a single function and does not target questions that would be resolved using a suite of functions. We leave this issue as a subject for future work. In Sects. 3.2–3.5, we explain each of the processes shown in Fig. 1.

3.2 Extracting the Function List

We extract a list of all of the functions for the target software application. This process must be performed manually if the target application does not provide a method for extracting such a list automatically. For research and development purposes, we targeted Word, Excel, and PPT in Microsoft Office 2000 experimentally, and extracted their function lists manually. We extracted all items in the drop-down menus and dialogue boxes as being functions of interest. This process is a monotonous task and therefore its manual cost is lower than those for compiling knowledge bases and dictionaries in existing QA systems [2], [3]. In our case, a human operator took a few seconds, on average, to identify a single function without any ambiguities.

Figure 2 shows one of the drop-down menus in Microsoft Word 2003. We extract the top-level items, such as “File” and “Edit”, and the lower-level items in the drop-down menus, such as “New” and “Close” in the “File” menu.

We maintain the hierarchy of these menu items, and use the complete description from the top-level menu item to the lowest-level item as a single function. We use the name of the target application as the topmost element. For example, the following function list was extracted from the

![Image](image.png)

**Fig. 2** A drop-down menu in Microsoft Word 2003.

data shown in Fig. 2.

- Word ⇒ File
- Word ⇒ File ⇒ New
- Word ⇒ File ⇒ New ⇒ Document
- Word ⇒ File ⇒ New ⇒ Template
  ···
- Word ⇒ File ⇒ Open
  ···
- Word ⇒ File ⇒ Close

Each function description in the list is an answer candidate.

Our method can be used for any software applications for which all the function names can be identified. This feature is independent of the user interface in which the functions are organized, such as drop-down menus and ribbon interfaces. If the functions are not organized in a hierarchy, each function description comprises only the names of the target application and the function, such as “Word ⇒ Open” and “Word ⇒ Close”, and our method is still available. In summary, although in this paper we targeted only applications in Microsoft Office 2000, our method can also be used for latest versions and even other applications as long as the list of the functions can be produced.

3.3 Collecting Web Pages

Given the function list extracted in Sect. 3.2, we use the description of each function as a query to search the Web. To produce the query for a function, all of the terms in the function description are combined with the “AND” operator. Because the name of a target application, such as “Word”, is included in the description of each function, pages related to that application are likely to be retrieved. We can use existing search engines, such as Google and Yahoo!, to collect Web pages. However, because the indexes of these search engines are updated periodically, it is difficult to reproduce our experimental results.

For experimental purposes, we use the NTCIR-5 Web collection [13], which comprises approximately 100 million pages extracted from the “.jp” domain. For indexing purposes, we remove the HTML tags from the documents.
and use ChaSen† to perform morphological analysis and to extract nouns, verbs, adjectives, out-of-dictionary words, and symbols as index terms. We use a variation of Okapi BM25 [14] as the retrieval model, which scores the relevance of a document to a query using Eq. (1). Instead of the inverse document frequency (IDF) factor used in the original Okapi BM25, \( \log \frac{N-n_t+0.5}{n_t} \), which takes a negative value when \( n_t \) is very large, we use the standard IDF:

\[
\sum_{k=1}^{N} \frac{(k_1+1) \cdot f_{i,d}}{(1-b)+b \cdot \frac{d_l}{\text{avgdl}}} \cdot \log \frac{N}{n_t} \frac{(k_3+1) \cdot f_{i,d}}{k_3+f_{i,d}}
\]

Here, \( f_{i,d} \) and \( f_{i,q} \) denote the frequency with which term \( t \) appears in document \( d \) and query \( q \), respectively. \( d_l \) denotes the length of document \( d \) in bytes, and \( \text{avgdl} \) denotes the average length of the target documents. \( N \) denotes the total number of target documents and \( n_t \) denotes the number of documents containing term \( t \). We empirically set \( k_1 = 2 \), \( b = 0.75 \), and \( k_3 = 1000 \).

3.4 Extracting Passages

The pages retrieved as described in Sect. 3.3 often contain fragments that are not related to the function in question. To discard any irrelevant fragments, such as advertisements, we extract passages that are highly related to the function. We segment each page into fixed-size (\( N \)) passages. We empirically set \( N = 500 \) bytes. We extract those passages in which a large number of query terms appear. However, because query terms can be split across passages, we produce extra overlapping segments, as shown in Fig. 3. For example, a fragment of Passage A is also included in Passage E.

Then we use the query to retrieve and sort related passages, for which we use the same indexing and retrieval methods as in Sect. 3.3. For each function, we retrieve up to ten of the top passages, discard the duplicated fragments, and expand the description of the function using the resulting passages. In Fig. 3, if Passages A and E are retrieved, then we discard the first half of Passage E to remove the redundancy. Through preliminary experiments, we determined the passage size and the number of top passages to be retrieved for each function.

3.5 Indexing and Retrieving Functions

We index both the original descriptions of the functions (see Sect. 3.2) and their expanded descriptions (see Sect. 3.4). We use the same indexing and retrieval methods as those described in Sect. 3.3. However, we use the two indexes to generate two independent ranked lists of functions for a single question. In each ranked list, each retrieved function is assigned a score. We compute the final score for a function \( f \) as \( S(f) \), defined in Eq. (2).

\[
S(f) = (1 - \alpha) \times S_o(f) + \alpha \times S_e(f)
\]

Here, \( S_o(f) \) and \( S_e(f) \) denote the score for \( f \) computed using the original and expanded indexes, respectively. The term \( \alpha \), which ranges from 0 to 1, is a parametric constant used to control the relative effects of \( S_o(f) \) and \( S_e(f) \) on the final score.

4. Experiments

4.1 Experimental Method

We evaluated the effectiveness of our QA method experimentally. We used Word, Excel, and PPT in Microsoft Office 2000 as the target applications. Although these are not the latest versions, many related pages for these versions are found in the NTCIR-5 Web collection, compared with those for newer versions. In addition, our method does not depend on the version of the target application.

To produce a test collection, we used two different sources from which we manually extracted questions that could be answered in terms of a single function and their correct answers. First, we used books of frequently asked questions (FAQs) for Microsoft products. Second, we used Moug††, a bulletin board system (BBS) concerning the operation of software. In Moug, questions are classified on an application-by-application basis. In total, we produced 353 questions.

Whereas experts edited the questions in the books, which are sometimes artificial, the questions in Moug are real-world questions. The question text in Moug often includes irrelevant fragments, such as greetings and typographical errors. We removed any greetings from question texts because a user would not input such greetings into a QA system. However, we retained other irrelevant fragments and used the entire text. Our preliminary experiments showed that the results were almost the same, irrespective of whether or not greetings were removed from the questions.

In the books and in Moug, a question usually comprises a concise title and a detailed description. We used both the titles and the entire texts as independent questions. However, in the PPT book, each question comprises a title alone. Example questions and their correct answers for Word are

\[\text{http://chasen.naist.jp/hiki/ChaSen/}\]
\[\text{http://www.moug.net/}\]
shown below, where <T> and <G> denote titles and greetings, respectively.

- Book

Q: <T>Printing out specific pages in a document</T> Can I specify the pages in a long document for printing out?
A: Word ⇒ File ⇒ Print ⇒ Print area

- BBS

Q: <T>Pop-up hints do not pop up</T> When I move a mouse pointer to a button on a toolbar, a pop-up hint used to pop up, but does not now. I have looked over several settings, such as “Tools ⇒ Options” and “Tools ⇒ Custom”, wondering if I could fix the problem. But I have not found which settings to configure. <G>I would appreciate it if you could help me. Best regards.</G>
A: Word ⇒ Tools ⇒ Custom ⇒ Options ⇒ Other ⇒ Display button names in toolbar

While the body of a question in the books is often a paraphrase of the title, in the BBS the body often includes additional fragments, such as backgrounds, experiences of a user, hypothetical solutions, and greetings. Although in the above BBS example, the user’s assumption is close to the solution, this is not always the case.

Table 1 shows the details of our test collection, in which “App”, “#Func” and “#Questions” denote the target applications, the number of functions for each application, and the number of questions, respectively. In Table 1, “AvgLen” denotes the average number of characters in the questions, with “Short” meaning only the title and “Long” meaning the entire question text, including the title but excluding greetings. In addition, “Avg#Ans” denotes the average number of correct answers and “AvgPath” denotes the average length of the path for correct answers from the top menu item, which is equal to the average number of menu items excluding the top item in correct answers. For example, the answer “Word ⇒ File ⇒ Close” includes two menu items. In Table 1, most of the values are given for “Book” and “BBS” independently.

From Table 1, we can see the characteristics of our test collection. Whereas each question for PPT was associated with a single answer, a number of questions for Word and Excel were associated with alternative answers. Alternative answers for a question can be either different functions or the same function reached via a different path in the menu hierarchy. Because each question in the PPT book comprises only a title, as described above, the average length of short PPT questions is the same as that of long PPT questions. According to “AvgPath”, a user has to navigate three to four correct menu items, on average, before arriving at a relevant function.

We manually categorized each question into three types with respect to the mismatch problem, namely “vocabulary mismatch”, “function mismatch”, and “none of these” (i.e., no mismatch problems occur). Examples of vocabulary mismatch include cases where the same concept was represented using different words in a question and its correct answer, such as “character shape/font style” and “slide design/slide layout”, and cases where users did not use concise technical terms, instead describing their needs in terms of verbose expressions. For example, the need “I do not want to allow other people to edit specific cells in Excel” can be satisfied by the function “protection”. An example of function mismatch is a user trying to change the color of input characters to black, whereas this problem can actually be resolved by changing the settings for “track changes”. When mismatch problems did not occur, users were often aware of relevant functions, but did not know where those functions are found within the menu hierarchy. In Sect. 4.2, we will use these categories to analyze the experimental results.

We used mean reciprocal rank (MRR) [5] as our evaluation measure. MRR has frequently been used to evaluate QA and precision-oriented retrieval. For each question, we calculated the reciprocal of the rank where the first correct answer was found. The MRR value is the mean of the reciprocal ranks for all questions and it ranges from 0 to 1.

We compared our method with three other types of method. First, the novelty of our QA method is that it expands the description of each function using related passages from Web pages. Therefore, one baseline method for comparison does not perform expansions, corresponding to our method with α = 0 in Eq. (2). We investigate how the value of MRR changes as the value of α increases.

Second, our method, which expands target functions, should also be compared with a method that expands queries using pseudo-relevance feedback (PRF) [15]. In PRF, top documents are collected in an initial retrieval and, from those documents, highly ranked terms are added to the original query. Each single document is the original description

---

The example questions and answers were derived from the Japanese versions of Microsoft products, and were translated into English for explanatory purposes. These translated answers are not necessarily the same as the answers derived from the English versions of Microsoft products.
of a function, which has not been expanded using its related passages. The rank of each term is determined by the value of Eq. (3), which is a product of term frequency (TF) and the IDF factors from Eq. (1).

\[
\frac{(k_1 + 1) \cdot f_{i,d}}{k_1 \cdot (1 + b \cdot \frac{d_l}{avgd}) + f_{i,d}} \cdot \log \frac{N}{n_i}
\]

(3)

Because additional terms can potentially introduce noise into the query, we multiply the value of Eq. (3) by a weight relative to the terms in the original query. The relative weight is a parametric constant used for all additional terms. If the relative weight is set to 1, the additional terms are considered as important as the original terms. However, the importance of additional terms decreases as the value of the relative weight decreases. In summary, PRF is associated with three parameters: the number of top documents in the initial retrieval \(nd\), the number of terms added to the original query \(nt\), and the relative weight of an additional term \(tw\).

The values for these three PRF parameters were optimized for the short and long questions independently. We changed the values for \(nd\) and \(nt\) from 10 to 100 in steps of 10, and changed the value for \(tw\) from 0.1 to 1 in steps of 0.1. The best MRR value for the short questions was obtained with \(nd = 70\), \(nt = 10\), and \(tw = 0.1\), and the best MRR value for the long questions was obtained with \(nd = 40\), \(nt = 10\), and \(tw = 0.1\).

Finally, we also compared our method with the help function embedded in each target application. Given a question, the help function presents a list of documents, each of which often provides one or more links mainly for reference purposes. To calculate the MRR value for the help function, an assessor submitted each test question to the help query box and identified the rank of the first document that described a correct answer. The assessor also identified whether or not a correct answer is described in documents linked directly from each ranked document.

Because the help function presented up to 20 documents for each question, we used up to 20 top-answer candidates in calculating the MRR value for all of the methods being compared. The baseline method and the method using PRF are lower-bound methods, and therefore our method should outperform them. However, because the help function presumably uses a number of dictionaries and knowledge bases, and has been tuned by human experts, including the help function in our tests will show the extent to which our straightforward method is comparable with a resource-intensive method for QA.

4.2 Results and Discussion

Table 2 shows the MRR values for all questions using different methods, in which the bold figure denotes the greatest MRR value for each question length (i.e., “Short” or “Long”). In Table 2, whereas \(\alpha = 0\) gives the MRR values for the baseline method, and positive values of \(\alpha\) give MRR values for our method with its Web-based expansion. In addition, “PRF” and “Help” give the MRR values for the PRF method and the help-function method, respectively.

For the help function, we evaluated MRR values for two different cases, depending on the definition of correct answers. For explanatory purposes, we use the term “ranked document” to refer to a document ranked in the list for a question and the term “linked document” to refer to a document linked directly from a ranked document. In Table 2, while “R” used correct answers found only in ranked documents to calculate MRR values, “RL” also used correct answers found in linked documents. Because linked documents are not ranked in a document list, unlike ranked documents, we approximated the rank of a linked document by the rank of the document linking to that document. However, because a user needs to read additional documents if a correct answer is in a linked document, MRR values for RL are somewhat inflated compared with the real-world utility.

From Table 2, we can see that, although the greatest MRR value was obtained for \(\alpha = 0.6\), the MRR values for our method with \(\alpha = 0.2\) and 0.4 were also greater than those for the baseline method (i.e., \(\alpha = 0\)), PRF, and the help function, irrespective of the question length. However, the MRR values for \(\alpha = 0.8\) and \(\alpha = 1\) were almost the same or even smaller than those for the other methods. The MRR value for PRF was smaller than that for \(\alpha = 0\), irrespective of the question length. This was probably because each document comprised a small number of terms, making effective terms relatively rare in the top documents in an initial retrieval. For most methods, the MRR value was increased when long questions were used. However, this is not the case for R and RL, where the MRR value for the short questions was always greater than that for the long questions.

We used the two-sided paired \(t\)-test for statistical testing to investigate whether the differences in Table 2 were meaningful or simply the result of chance occurrence [16]. Table 3 shows the results, in which “<” (or “>”) and “<<” indicate that the difference between two results was significant at the 5% and 1% levels, respectively, and “—” indicates that the difference between two results was not significant. From Table 3, we can see that our method with \(\alpha = 0.2, 0.4\), and 0.6 outperformed significantly the baseline.
method, PRF, and the help function with respect to MRR, for both question lengths. In some cases, our method with \(\alpha = 0.8\) and \(\alpha = 1\) also outperformed significantly the help function. However, because the baseline method outperformed significantly our method with \(\alpha = 1\), the preferred value for \(\alpha\) should be between 0.2 and 0.6. In summary, we conclude that the Web-based expansion method is effective in improving the MRR of QA for software applications.

Table 4 shows the distribution of the ranks at which the first correct answer was found. In Table 4, “NA” denotes the number of questions for which correct answers were not found in the document list. Because the help function presents up to 20 documents, its values for NA are large. However, for the other methods, the value of NA is not found in the question list. As explained in Sect. 4.1, we used up to 20 top-answer candidates in calculating the MRR value for all the methods being compared. In other words, for the questions associated with “21–50”, “51–100”, “101–”, and “NA”, we considered that no correct answer was found.

From Table 4, the Web-based expansion method increased the number of questions for which the first correct answer was found in a high rank. For the short questions, when \(\alpha\) was set to a value of 0.6, the top candidate was the correct answer for 19.0% of the questions, and the correct answer was found in the top 10 candidates for 36.3% of the questions. For the long questions, these ratios were 19.8% and 37.7%, respectively. However, there were many questions for which the rank of the first correct answer was over 100. This is mainly caused by correct answers being expanded with irrelevant terms and incorrect answers being expanded with terms related to a question. We intend to improve the selection of related terms used for expansion purposes in the future.

Tables 5–7, which use the same notation as Table 2, show the evaluation of our method from different perspectives.

Table 5 shows the MRR values for each of the three target applications. Whereas the best MRR value was obtained for \(\alpha = 0.6\) in most cases, the best MRR value for the short PPT questions was obtained by the baseline method, and the best MRR value for the long Word questions was obtained for \(\alpha = 0.4\). However, across both short and long questions, the best MRR value for PPT was obtained for \(\alpha = 0.6\). Therefore, the Web-based expansion method appears effective for any target application.

Table 6 shows the MRR values for each of the two question sources. The best MRR value was usually obtained for \(\alpha = 0.6\), whereas the best MRR value for the short questions in the books was obtained for \(\alpha = 0.4\). Therefore, Web-based expansion was effective for both expert and non-expert questions. However, as predicted, the MRR values for the questions from the BBS were generally worse than those for the book-related questions. One solution would be to discard or organize the irrelevant fragments that are often included in real-world questions. However, as explained in Sect. 4.1, the results were almost the same with and without

```
<table>
<thead>
<tr>
<th>α</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
<th>0.8</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRF</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>---</td>
<td>&gt;</td>
</tr>
<tr>
<td>R</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>RL</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Long</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRF</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>R</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>RL</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
```

Table 3: t-test results of the differences between QA methods ("<": 0.01, "<": 0.05, ">": 0.05, "—": not significantly different).

```
<table>
<thead>
<tr>
<th>Rank</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1–5</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Short</td>
<td></td>
</tr>
<tr>
<td>PRF</td>
<td>59</td>
</tr>
<tr>
<td>R</td>
<td>56</td>
</tr>
<tr>
<td>RL</td>
<td>20</td>
</tr>
<tr>
<td>Long</td>
<td></td>
</tr>
<tr>
<td>PRF</td>
<td>55</td>
</tr>
<tr>
<td>R</td>
<td>55</td>
</tr>
<tr>
<td>RL</td>
<td>22</td>
</tr>
</tbody>
</table>
```

Table 4: Distribution of the ranks at which the first correct answer was found.
Table 5  MRR values for each application.

<table>
<thead>
<tr>
<th>α</th>
<th>PRF</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.1935</td>
<td>.1210</td>
</tr>
<tr>
<td>.2</td>
<td>.2111</td>
<td>.1485</td>
</tr>
<tr>
<td>.4</td>
<td>.2171</td>
<td>.2143</td>
</tr>
<tr>
<td>.6</td>
<td>.2359</td>
<td>.2322</td>
</tr>
<tr>
<td>.8</td>
<td>.1864</td>
<td>.1404</td>
</tr>
<tr>
<td>1</td>
<td>.1210</td>
<td>.1404</td>
</tr>
</tbody>
</table>

Short E .2230 .2268 .1442 .1404 .1814
P .2243 .2177 .1485 .1404 .1883
W .1935 .2359 .1864 .1210 .1723

Long E .2230 .2494 .2378 .2268 .2222
P .2243 .2197 .1485 .1404 .1883
W .1935 .2359 .1864 .1210 .1723

#Questions: Word (W) = 121, Excel (E) = 148, PowerPoint (P) = 84

Table 6  MRR values for each question source.

<table>
<thead>
<tr>
<th>α</th>
<th>PRF</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.2986</td>
<td>.2912</td>
</tr>
<tr>
<td>.2</td>
<td>.3230</td>
<td>.3107</td>
</tr>
<tr>
<td>.4</td>
<td>.3555</td>
<td>.3486</td>
</tr>
<tr>
<td>.6</td>
<td>.3348</td>
<td>.3348</td>
</tr>
<tr>
<td>.8</td>
<td>.2966</td>
<td>.2966</td>
</tr>
<tr>
<td>1</td>
<td>.2393</td>
<td>.2393</td>
</tr>
</tbody>
</table>

Short Book .2986 .2912
BBS .0818 .0869
W .2986 .2912

Long Book .3033 .2923
BBS .0900 .0870
W .3033 .2923

#Questions: Book = 214, BBS = 139

Table 7  MRR values for each question type.

<table>
<thead>
<tr>
<th>α</th>
<th>PRF</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.1366</td>
<td>.1336</td>
</tr>
<tr>
<td>.2</td>
<td>.1442</td>
<td>.1110</td>
</tr>
<tr>
<td>.4</td>
<td>.1704</td>
<td>.1358</td>
</tr>
<tr>
<td>.6</td>
<td>.2112</td>
<td>.1756</td>
</tr>
<tr>
<td>.8</td>
<td>.1955</td>
<td>.1756</td>
</tr>
<tr>
<td>1</td>
<td>.1756</td>
<td>.1756</td>
</tr>
</tbody>
</table>

Short Vocab .1366 .1336
Func .0613 .0529
None .3372 .3265
W .1366 .1336

Long Vocab .1397 .1325
Func .0612 .0574
None .3483 .3378
W .1397 .1325

#Questions: Vocab = 93, Function (Func) = 91, None = 169

greeting removal. This implies that a more sophisticated approach to irrelevance is needed. We leave this as a subject for future work.

Table 7 shows the MRR values for each of the three question types, in which “Vocab”, “Func”, and “None” denote the MRR values for the vocabulary mismatch, the function mismatch, and neither of these, respectively. The number of questions for Vocab, Func, and None were 93, 91, and 169, respectively. In our test collection, approximately half of the questions were associated with mismatch problems in which vocabulary and function mismatch contributed almost equally.

Although the best MRR value was usually obtained for α = 0.6, there were two exceptions. First, for the short questions in Func, the best MRR value was obtained for RL, and this value was noticeably greater than those for the other methods. In other words, reference links in the help function were effective in alleviating the function mismatch problem. However, even in this case, the MRR values for our method with α = 0.2, 0.4, 0.6, and 0.8 were greater than those for both the baseline method and PRF. In addition, as explained earlier, the MRR values for RL are inflated. Second, for the long questions in None, the best MRR value was obtained for α = 0.4. Overall, therefore, the Web-based expansion method is effective for any type of question.

For Vocab and Func, the MRR values for α = 0.8 were also greater than those for α = 0, unlike the results shown in Tables 2, 5, and 6. However, this tendency was not observed for None. As predicted, the expansion was especially effective when either of the mismatch problems occurred. In addition, the MRR values for Func were generally smaller than those for Vocab and None. We aim to reduce inaccuracy caused by the function mismatch problem in future work.

Throughout our experiments, the MRR values for α = 1, in which only the index for the expanded function descriptions was used, were lower than those for other values of α. This showed the validity of using the original and expanded indexes together. However, it is also important to enhance the expanded index itself. For this purpose, we need more sophisticated term weighting for the expansion. Although we used a TF.IDF-style method for term weighting, a number of methods for measuring the semantic similarity between words [12] should also be used in future work.

5. Conclusion

Reflecting the rapid growth of information technology, the configuration of software applications such as word processors and spreadsheets is both sophisticated and complicated. It is often difficult for users to identify relevant functions in the online manual for a target application. In this paper, we have proposed a QA method for software applications that finds functions related to the user’s request. To enhance our
method, we addressed two “mismatch” problems, one associated with mismatch of vocabulary, and the other associated with mismatch of function.

To resolve these mismatch problems, we expanded the description of each function, allowing a user’s questions to be matched to relevant functions with high accuracy. We proposed a Web-based method for expansion purposes, and we used the description of each function as a query to search the Web for related pages. We then extracted any related passages from the retrieved pages. The description of each function was expanded by including the corresponding passages. We indexed both the original descriptions and the expanded descriptions. The resulting two indexes were used to retrieve functions related to user questions.

For evaluation purposes, we targeted Microsoft Word, Excel, and PPT, and produced a test collection comprising 353 questions extracted from FAQ books and a BBS, along with their correct answers. Our method was compared with a method with no expansion, pseudo-relevance feedback, and the help function embedded in each application. The experimental results showed that the Web-based expansion method was effective in enhancing QA for software applications. At the same time, the accuracy of our method was relatively low for questions in the BBS and those associated with the function mismatch problem. Future work will include improving the overall accuracy of our method and the targeting of questions for which a suite of functions is needed to provide answers.

Acknowledgments

This study was partly supported by Grant-in-Aid Scientific Research on Priority Area of “Cyber Infrastructure for the Information-explosion Era” (Grant No.18049006, 19024007, and 21013003).

References


Atsushi Fujii  received his Ph.D. degree in  engineering from the Graduate School of  Information Science and Engineering at Tokyo Institute of Technology in 1998 and is currently an associate professor of the Graduate School of  Information Science and Engineering at Tokyo Institute of Technology. His research interests are natural language processing, information retrieval, and Web mining.

Seiji Takegata  received his B.E. from the Faculty of Engineering Division at Tokyo University of Science in 1988 and M.S. from the Graduate School of Library, Information and Media Studies at University of Tsukuba in 2007. He currently runs a business in documentation for computer systems at Veltex Co., Ltd.