Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Analyzing close relations between target artifacts for improving IR-based requirement traceability recovery*

Haijuan WANG[†1], Guohua SHEN[†‡1,2,3], Zhiqiu HUANG[†1,2,3], Yaoshen YU[†1], Kai CHEN[1]

[1]*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China*

[2]*Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 211106, China*

[3]*Key Laboratory of Safety-Critical Software, Ministry of Industry and Information Technology, Nanjing 211106, China*

[†]E-mail: 18895681787@163.com; ghshen@nuaa.edu.cn; zqhuang@nuaa.edu.cn; yaoshen.yu@outlook.com

Received Mar. 26, 2020; Revision accepted June 23, 2020; Crosschecked June 8, 2021

Abstract: Requirement traceability is an important and costly task that creates trace links from requirements to different software artifacts. These trace links can help engineers reduce the time and complexity of software maintenance. The information retrieval (IR) technique has been widely used in requirement traceability. It uses the textual similarity between software artifacts to create links. However, if two artifacts do not share or share only a small number of words, the performance of the IR can be very poor. Some methods have been developed to enhance the IR by considering relations between target artifacts, but they have been limited to code rather than to other types of target artifacts. To overcome this limitation, we propose an automatic method that combines the IR method with the close relations between target artifacts. Specifically, we leverage close relations between target artifacts rather than just text matching from requirements to target artifacts. Moreover, the method is not limited to the type of target artifacts when considering the relations between target artifacts. We conduct experiments on five public datasets and take account of trace links between requirements and different types of software artifacts. Results show that under the same recall, the precisions on the five datasets improve by 40%, 8%, 20%, 4%, and 6%, respectively, compared with the baseline method. The precision on the five datasets improves by an average of 15.6%, showing that our method outperforms the baseline method when working under the same conditions.

Key words: Requirement traceability; Information retrieval; Close relations; Target artifacts
https://doi.org/10.1631/FITEE.2000126 **CLC number:** TP311.5

1 Introduction

Requirement traceability (RT) is defined as "the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)" (Gotel and Finkelstein, 1994). As a research direction of RT, traceability link recovery is a software engineering task that deals with the identification and comprehension of dependencies and relations between software artifacts.

As an important link in software process management, RT plays a significant role in ensuring system quality and responding to changing requirements. With RT, software developers can discover dependencies between artifacts, assess requirement coverage, and calculate the impact of requirement

‡ Corresponding author

* Project supported by the National Key Research and Development Program, China (No. 2018YFB1003902), the National Natural Science Foundation of China (No. 61772270), and the Funding of the Key Laboratory of Safety-Critical Software (No. 1015-XCA1816403)

ORCID: Haijuan WANG, https://orcid.org/0000-0002-2088-3563; Guohua SHEN, https://orcid.org/0000-0003-2182-0019

changes. On one hand, RT is dedicated to helping software developers do tracing analysis to determine whether all low-level elements (such as design and source code) have corresponding requirements. On the other hand, it can be used for integrity analysis and test coverage evaluation to determine whether all requirements have been implemented and tested accordingly (Hayes et al., 2005).

In this paper, we focus on the linkage between requirements and various artifacts (such as use case, design, and test case) to provide more support for software development and maintenance activities.

With the increasing size and complexity of software systems, manual recovery and maintenance of trace links are time-consuming and laborious. The advantage of using information retrieval (IR) is that it can automatically recover trace links through text analysis. This is very helpful for solving traditional requirement tracing problems such as heavy manual workload, difficult maintenance, and being error-prone. However, a basic problem of IR is that the textual similarity calculation is based on the keyword matching degree between two kinds of artifacts. If the two artifacts (e.g., the requirement and the source code) are heterogeneous, they may not share a common vocabulary, synonyms, or the language structure, and IR will miss the link.

Automatic tracing methods in this field have been studied a lot in recent years. For example, strategies such as thesaurus utilization (Wang et al., 2009; Leuser and Ott, 2010), project glossary (Zou et al., 2010), phrasing (Zou et al., 2006, 2010), key phrases (Chen and Grundy, 2011), and refactoring (Mahmoud and Niu, 2013, 2014) have been used to reduce the adverse effects caused by inconsistent terminology or missing, misplaced, and duplicated signs in textual artifacts. Nevertheless, these strategies were used mainly for text processing of artifacts, ignoring the consideration of the relations between target artifacts.

Trace links exist widely between the requirements and different types of software artifacts. These artifacts contain, but are not limited to, software architecture documents, design models, source code, test cases, and configuration files. For example, understanding which requirements are covered by test cases, i.e., requirement tracing in test cases, is an important concern in the quality assurance of complex software systems. Capturing the traceability relationship between software requirements and design allows developers to check whether the design meets the requirement and to analyze the impact of requirement changes on the design. Some methods are often limited to certain types of software artifacts or require specific preconditions. Therefore, developers cannot easily apply them to general use. This makes it a bit difficult for developers to decide which method can be adopted for their projects.

In considering the relations between the target artifacts to enhance IR, the focus is on the code (Panichella et al., 2013; Kuang et al., 2017). The reason is that code is a structured document, and there are some relations (e.g., inheritance, aggregation, and dependencies) among code. However, few people analyze the relations between other types of target artifacts, such as test cases. Since there are dependencies between the code that can be used, considering relations between other types of target artifacts (such as test cases) can also be used. Therefore, we develop a method that has few limitations and minimize the impact of diversity of target software artifacts. That is our contribution.

In this paper, we present a novel method by using close relations between target artifacts to improve IR-based traceability link recovery. To the best of our knowledge, there is no research to analyze the relations between target artifacts other than code to improve the IR model in RT tasks. First, we use the IR method to create candidate links. Then, we apply the "bonus" value to some trace links, and the target artifacts in these trace links are closely connected with the target artifacts in the correct trace links. We apply this method to five open datasets, and experimental results show that the precisions on EasyClinic, CM1-NASA, Pine, GANNT, and iTrust are increased by 40%, 8%, 20%, 4%, and 6%, respectively, with the same recall value. The precision on the five datasets improves by 15.6% on average, showing that this method can effectively improve the performance of the IR model.

2 Background

2.1 IR-based method

IR refers to a type of technique used to compute textual similarities among different documents. The textual similarity is computed using the

occurrence of terms in the documents. If two documents share a large number of terms, those documents are considered to be similar. There are a number of IR methods studied in the automated tracing literature, including the vector space model (VSM) (Hayes et al., 2006), latent semantic indexing (LSI) (Marcus and Maletic, 2003), and probabilistic model (PM) (Antoniol et al., 2002). In the VSM, each document is represented as a vector of terms. The similarity of two documents is calculated based on the angle between each document's vector. For example, the similarity may be the cosine of the angle between the vectors. LSI determines the degree of correlation by analyzing the implicit "semantic correlation" between texts. This method improves the vector space model to capture not only the semantics of individual words, but also the semantics of sentences, paragraphs, and the whole literature, instead of focusing on the original literal word description. The PM uses document and query related probabilities to calculate document similarity and query similarity, respectively.

In conclusion, although these algorithms can automatically analyze the text, establish the candidate trace links, and reduce human labor, there are a large number of natural language descriptions in the software artifacts; that is, accurate lexical, syntax, and semantics are still facing huge challenges. Therefore, how to improve the IR method has become the focus of research.

2.2 CRT-based method

In software engineering, there is an association between target artifacts (such as use cases) derived from the same requirement. For some textual artifacts (with a certain form of structure), such as source codes and use cases, as we know, there are relations of inheritance and aggregation among source codes that implement the same requirement, and relations of generalization and inclusion among use cases that implement the same requirement. All of these textual types of target artifacts can be preprocessed to convert them to the natural text form. Moreover, the relation between the target artifacts does not disappear with the pre-processing. The associations that still exist between the various types of target artifacts are collectively referred to as close relations. In fact, we can think of this close relation as representing a semantic connection between the two target artifacts.

Later, we also calculate the semantic similarity between the two target artifacts to indicate whether there is a close relation between them. In other words, there is a close relation between the target artifacts that implement the same requirement. Our goal is to discover and use the close relations between the target artifacts (called the CRT-based method) to contribute to the tracing task.

Embedding (also known as distributed representation) is a technique for learning vector representations of entities (such as words, sentences, and images), in which similar entities have vectors that are close to each other. Word embedding can be used to convert a word into a fixed-length vector representation for easy mathematical processing. Words with similar contexts have similar meanings and similar vector representations. The word embedding technique has been widely used to solve the semantic similarity matching problem. Combining the IR technique with word embedding techniques, the candidate list obtained from IR is re-ranked after considering the close relations between target artifacts. This can effectively improve IR-based requirement traceability.

3 Approach

In this section, we describe a method for trace link recovery combining the IR method and close relations between target artifacts (called IR_CRT). We conjecture that the close links in target artifacts are transitive for traceability recovery. If a close link exists between two target artifacts, and a textual link (from an IR method) exists between a source artifact and one of the target artifacts, then we conjecture that a link exists from the source artifact to both target artifacts. In practice, a method combining IR and semantic information detects the transitive relations and assigns a bonus to the textual similarity of those relations (Panichella et al., 2013).

The traceability L is expressed as a set of trace links L= $\{l_1, l_2, \cdots, l_n\}$, and each trace link l_i is a three-tuple $\langle$src, tgt, sim$\rangle$, where src is a source artifact, tgt is a target artifact, and sim is the similarity degree between src and tgt. The requirement region identifies all target artifacts that trace to a given requirement. Fig. 1 is a description of requirement traceability and the requirement region.

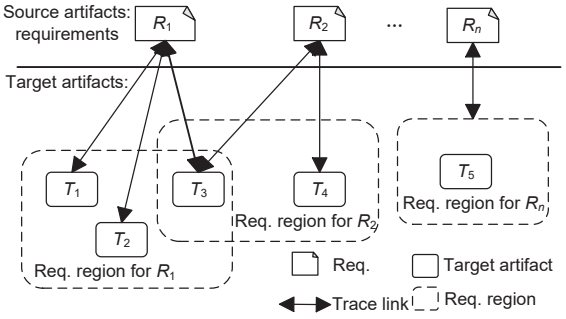

Fig. 1 Requirement traceability and requirement region

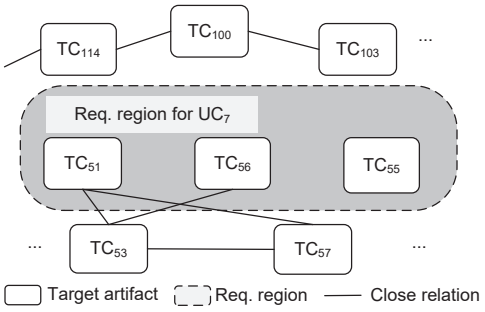

Fig. 2 Excerpt of CRTG for EasyClinic (requirement region for UC_7 is highlighted in gray)

3.1 CRT graph and requirement region

We represent the close relations between target artifacts of a software system as a close graph (CRTG). When the close relations between our target artifacts have been calculated, formally, let $G(T, E)$ be the graph of close relations between target artifacts, where $T = \{t_1, t_2, ..., t_n\}$ is the set of nodes (target artifacts) and $E = \{e_1, e_2, ..., e_m\}$ is the set of edges. Each edge $(t_i, t_j) \in E$ represents a close relation. Let $S = \{s_1, s_2, ..., s_p\}$ be the set of source artifacts (e.g., use cases) and $L = \{l_1, l_2, ..., l_k\}$ the set of links. Each link $(s_i, t_j) \in L$ ($s_i \in S$ and $t_j \in T$) represents a candidate link obtained by IR. Fig. 2 depicts an excerpt of the CRTG of an EasyClinic system, which is provided by the CoEST community (http://www.coest.org/). Research has found that requirements are usually implemented in connected areas of a CRTG and these connected areas are called requirement regions (Burgstaller and Egyed, 2010). The requirement region (highlighted in gray in Fig. 2) identifies all test cases (TC) that implement the requirement UC_7 (here UC means use case) of the EasyClinic system.

If there is a link between the target artifact T_1 and the source artifact S_1, several of the close (most semantically similar) target artifacts to the target artifact T_1 (in CRTG, close to T_1) will be given an additional bonus, and the similarity value will be recalculated to make these target artifacts more traceable to S_1. For example, in Fig. 2, there is a link between the target artifact TC_{51} and the source artifact UC_7; the similarity values between TC_{53} and UC_7, and between TC_{57} and UC_7 related to the semantics of TC_{51} will be recalculated to improve the ranking of the correct link (TC_{53}, TC_{57}) in the candidate link. Conversely, all target artifacts with the closest semantic relations to the target artifact are not traceable when the target artifact has a high possibility of not tracing to the source artifact. The target artifact TC_{100}, in Fig. 2, is an example.

3.2 Candidate list generation and re-ranking

Our method consists mainly of three steps (Fig. 3).

First, a traditional IR-based traceability recovery method is used to generate an original candidate list of traceability for each requirement R_i. To recover trace links with IR-based approaches, we perform a few standard document pre-processing steps (normalizing), such as removing stop words (like "a," "an," and "the" that have no real meaning), punctuation, and particular words (unique to a dataset, such as EasyClinic). When the type of target artifact is code, some identifiers named according to the camel's hump rule are also cut into separate words. To eliminate the differences between the writing styles of source and target artifacts, CoreNLP is used to extract the word stem and restore the word form. The pre-processing code snippet is shown in Fig. 4. These pre-processing steps also provide support for the subsequent use of word embedding to calculate the similarity between the target artifacts. Then, we use the VSM to calculate the similarity values of the source artifact and the target artifact to generate the candidate trace link. These commonly used algorithms often replace each other; however, in terms of ease of use, the VSM is used. Therefore, this experiment is based on the VSM to obtain the candidate list. Here, we use RETRO (https://opensource.gsfc.nasa.gov/projects/RETRO) (Hayes et al., 2007), a tool that integrates VSM algorithms, to generate a list of tracing candidates.

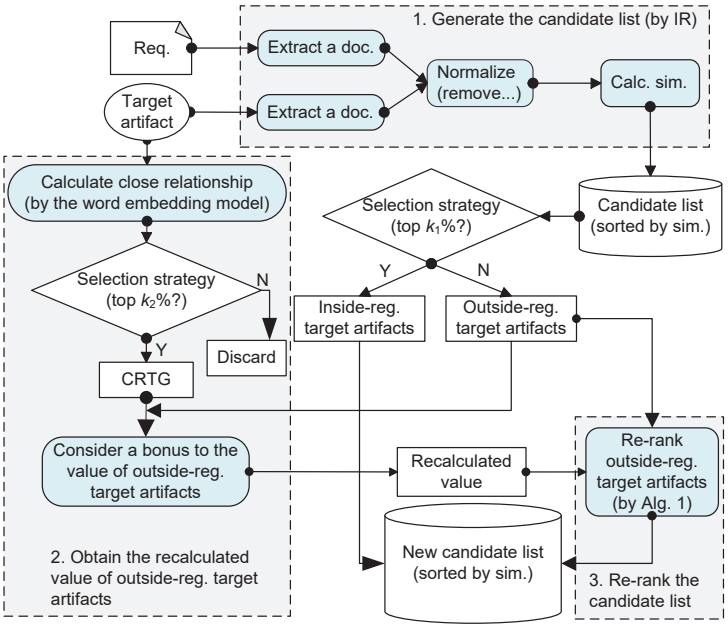

Fig. 3 Candidate list generation and re-ranking process

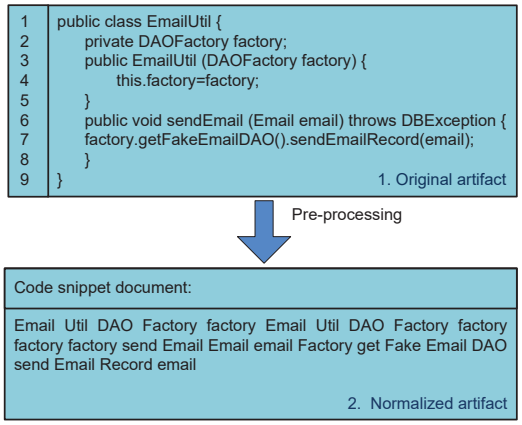

Fig. 4 Data pre-processing demonstration

Second, for each R_i, we generate a candidate list cl through the IR-based traceability recovery method. The target artifacts in cl are sorted according to the similarity values from high to low. We need to cut the ranked candidate list cl to form the initial requirement region. Here, we select the first cut percentage needed for the experiment. This is the cut percentage of the requirement region. Usually, cut-point or threshold based strategies can be adopted to cut the list. The first category cuts the ranked list regardless of the similarity values by using a constant k or a variable k% (called cut percentage). The second category uses a threshold ξ on a similarity value and only the pairs of artifacts having a similarity value greater than or equal to ξ will be retrieved. Constant threshold (e.g., ξ=0.70) or scale threshold (i.e., $\xi=c\cdot$MaxSimilarity, where $0\leq c\leq 1$) is commonly used for this strategy. Each outside-region target artifact (the target artifact outside the initial requirement region) is obtained through the inside-region target artifacts (the target artifacts inside the initial requirement region) and the CRTG.

Word embedding is a technique that can be used for computing semantic similarity between texts, and it can be used for estimating whether there is a close relation between target artifacts in a project. A well-behaved word embedding model often needs a large number of corpus-related fields to train it. Given that our target artifacts cover many areas, such as medicine and aeronautics, we use generic domain word embedding models, such as word2vec. The main idea of word2vec is to train a model according to the context of each word so that similar words will have similar numerical representations. The word2vec embedding model used here was trained using the Google Negative News text corpus (about 100 billion words, containing 300-dimensional vectors for three million words and phrases). It is a general domain corpus. Because of the limitation of datasets, no domain-specific corpus can be

obtained. In the above work, we have pre-processed the datasets, taking the pre-processed datasets as the input of the embedding model, and the output is the similarity value between target artifacts. For each T_j, through word embedding, we can obtain the close relation list sl ranking from high to low in similarity value. For T_j, we assume that the top $k\%$ links of its close relation list are closely related to T_j. We also need to cut the ranked candidate list sl to form the CRTG. Here, we select the second cut percentage needed for the experiment, which is the cut percentage of the CRTG.

Third, for each R_i, inside-region target artifacts are directly added to a new candidate list, and the outside-region target artifacts are re-ranked (Algorithm 1) by considering whether to increase a bonus. Then we add the candidate trace links with recalculated similarity value to the new candidate list. The size (measured as the difference between the maximum and minimum similarity values) of the ranked list can sensibly differ from one system to another, or when tracing different types of artifacts. For this reason, we propose an adaptive bonus that is proportional to the median variability of the similarity values computed for each software artifact. More precisely, we set the adaptive bonus as $\delta = \text{median}\{v_i, v_{i+1}, ..., v_n\}$ where a generic v_i value denotes the variability of the similarity value of the i^{th} artifact, i.e., $v_i = (\max_i - \min_i)/2$, where $\max_i$ and $\min_i$ are the maximum and minimum similarity values of the i^{th} source artifact, respectively (Panichella et al., 2013). With respect to the constant bonus, the proposed approach takes into account the variability of the ranked list. The new candidate list is a re-ranking of the original candidate list.

Algorithm 1 Re-ranking the original candidate list

Input: Original candidate list
Output: Re-ranked candidate list

```
i ← 1 // i means the position of a trace link in the
// candidate list
while not (end of List) do
    Obtain the top k% links (s, t_j) in position i of
    List
    for all t_p ∈ T do
        if (t_j, t_p) ∈ E then
            Sim(s, t_p) ← Sim(s, t_p) + δ· Sim(s, t_p)
            // Each edge (t_i, t_j) ∈ E represents a close
            // relation
        end if
    end for
    i← i + 1
end while
Re-ranked List
```

4 Evaluation

To find out whether we can improve the efficiency of the IR method by leveraging close relations between target artifacts, we propose the following research questions:

RQ1: What is the best setting for our method (such as the size of the initial requirement region)?

RQ2: Whether our method is generic to create links from requirements to x (x can be test cases, designs, code, etc.) without considering what type of artifact the target is?

To answer these questions, we design two groups of experiments. Table 1 shows the characteristics of the considered software systems in terms of type and number of source/target artifacts. Table 1 also reports the number of correct links between different types of source and target artifacts. Such information (that we use as an oracle to evaluate the accuracy of the proposed traceability recovery methods) is derived from the traceability matrix provided by the original developers.

To respond to the first research question, we undertake two steps. First, we need to select a dataset at random, and then use the RETRO tool (using VSM with TF-IDF weighting) to generate candidate trace links for artifact pairs. Second, we need to set the two cut percentages required by our program as 1%, 3%, 5% of the original requirement region and 5%, 10%, 15% of the CRTG respectively, and then conduct random combination to generate a total of nine sets of experimental results. For the second research question, we also undertake two steps. First, as mentioned earlier, candidate links are obtained using the IR method. Second, according to the results of the first experiment, the best setting of the method is selected for the experiment.

The performance of IR-based traceability recovery methods is usually measured using two metrics, recall and precision. Recall is the ratio of the number of links that are successfully retrieved to the number of links that are relevant. Precision is the ratio of the number of links that are successfully retrieved to

Table 1 System information

System	Source artifact (number of source artifacts)	Target artifact (number of target artifacts)	Number of correct links
EasyClinic	UC (30)	TC (63)	63
CM1-NASA	Req. (235)	Design (220)	353
Pine	Req. (49)	UC (51)	250
GANNT	HR (17)	LR (69)	68
iTrust	UC (36)	CC (106)	174

UC: use case; TC: test case; HR: high-level requirements; LR: low-level requirements; CC: class code

the number of all retrieved links (Armstrong, 2013):

$$\text{Recall} = \frac{|\{\text{relevant\_links}\} \cap \{\text{retrieved\_links}\}|}{|\{\text{relevant\_links}\}|}, \tag{1}$$

$$\text{Precision} = \frac{|\{\text{relevant\_links}\} \cap \{\text{retrieved\_links}\}|}{|\{\text{retrieved\_links}\}|}. \tag{2}$$

To further measure the overall quality of the experimental results, we select another two commonly used metrics: AP (average precision) and MAP (mean average precision) (Dietrich et al., 2013). AP measures how well relevant documents of all queries (requirements) are ranked to the top of the retrieved links, and it is computed as follows:

$$\text{AP} = \frac{\sum_{r=1}^{N} \text{Precision}\,(r) \cdot \text{isRelevant}\,(r)}{|\text{RelevantDocuments}|}, \tag{3}$$

where r is the rank of the target artifact in a ranked list of links, isRelevant() is a binary function assigned "1" if the link is relevant and "0" otherwise, Precision() is the precision computed after truncating the list immediately below that ranked position, and N is the total number of documents. When multiple links are listed for a single similarity value (e.g., at similarity of zero), the links are evenly distributed across the space of that score, simulating their random distribution. MAP is the mean of the AP scores over a set of queries (requirements), and it is computed across all queries as follows:

$$\text{MAP} = \frac{\sum_{q=1}^{Q} \text{AP}}{Q}, \tag{4}$$

where q is a single query and Q is the total number of queries.

5 Results

The method mentioned in Section 3 is implemented and several common metrics in IR, like precision, recall, and MAP, are used for evaluation.

5.1 RQ1—determining the best setting of the IR_CRT method

We need a setting for our method, such as the size of the initial requirement region (the value for variable k%). Our method is called IR_CRT (k_1% & k_2%), where IR indicates the underlying information retrieval method, and k_1 and k_2 are the cut percentages for requirement region and CRTG respectively. For example, the method IR_CRT (1% & 1%) means that the IR method is adopted, the initial requirement region is the top 1% of the candidate list generated by IR, and the CRTG is the top 1% of the close relation candidate list by word2vec. Here, VSM is used as the underlying IR method to automatically generate trace links for the dataset (EasyClinic). Figs. 5 and 6 show the precision–recall curves and MAP values at different values of cut-point variable (k_1% & k_2%) on EasyClinic, respectively. The figures show that the results obtained by our method are superior to those of IR regardless of the combination of the cut percentages for our method. However, to verify the validity of our method on different datasets, we select the best performing combination, namely, 5% & 10%. Hereafter, (5% & 10%) is adopted for further experiments. Generally, the setting of this method depends on the size and quality of datasets. Although some cut percentages can be recommended, we need to conduct experiments to tune the cut percentages for the best performance in different situations.

5.2 RQ2—verifying that the IR_CRT method is not limited to the target artifact type

In the second experiment, we add four different types of datasets of target artifacts (low-level requirements, code, design) and compare our method (IR_CRT (5% & 10%)) with the IR method. For

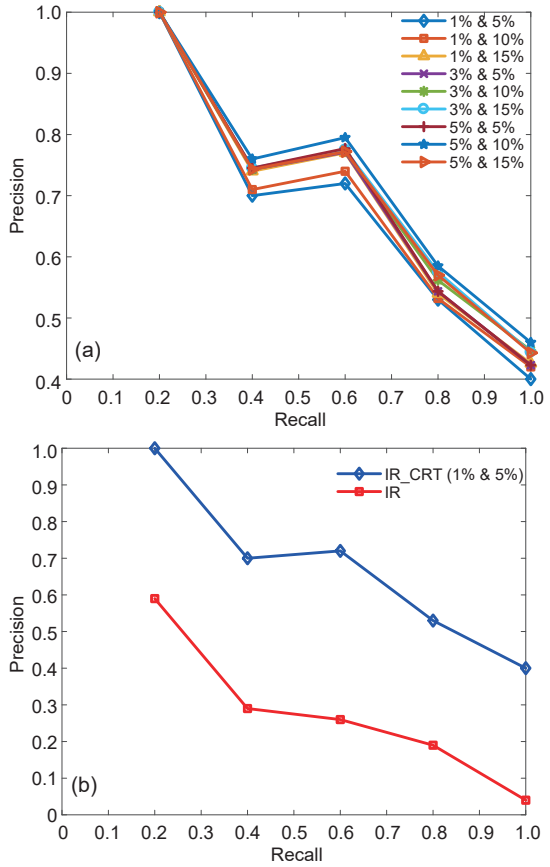

Fig. 5 Precision–recall curves on EasyClinic: (a) IR_CRT at different cut percentage values; (b) IR_CRT (1% & 5%) versus IR

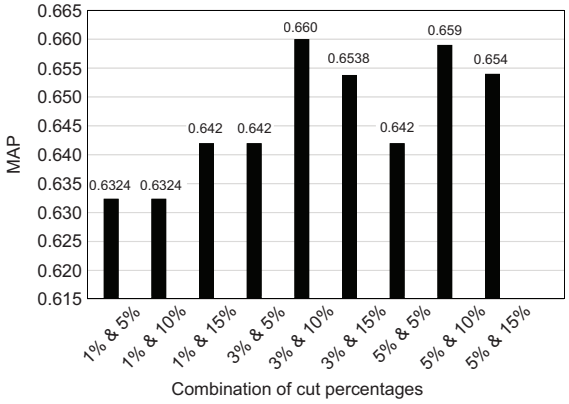

Fig. 6 MAP values of IR_CRT at different cut percentage values on EasyClinic

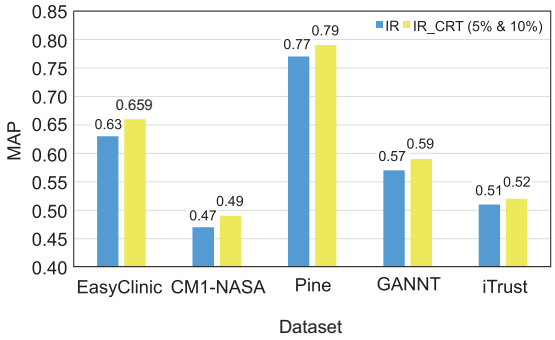

Fig. 7 MAP values of IR versus IR_CRT (5% & 10%) on different datasets

the sake of fairness, these two methods adopt the same IR method VSM. Fig. 7 shows the comparison of MAP values between IR_CRT and the baseline method. Fig. 8 shows the comparison of the precision–recall curves between IR_CRT and IR. Specifically, from Fig. 7, the ranking quality obtained by the IR_CRT (5% & 10%) method is superior to that by the baseline method. For the GANNT dataset, the MAP values of IR_CRT (5% & 10%) and IR are 0.59 and 0.57, respectively. From Fig. 8, it can be seen that the highest value of the IR_CRT precision improvement is 39.24% (at the 100% recall on the Pine dataset) compared with IR.

For the iTrust dataset, its target artifact is code, and then we reproduce the O-CSTI (optimistic combination of structural and textual information) (Panichella et al., 2013). O-CSTI (the target artifact can only be code) uses an IR method to locate a set of initial trace links, and then extends that set using the structural information. However, only structural information is considered when the traceability links from the IR method are verified by developers and classified as correct links. Although this method can improve accuracy, it requires extra manpower and degrades the automation of the tool. Moreover, from the experimental results of iTrust, our method performs better.

As we can see, this method improves less on the iTrust dataset than the other four. The core of this method is to improve the initial candidate list by finding the close relations between the target artifacts. The word embedding technique is used to calculate the close semantic relations between target artifacts; however, for different types of target artifacts, the calculation results of word embedding are different.

The word embedding technique usually performs well when it is used to calculate similarity between textual documents. Compared with code (with some syntax and structure), the word embedding technique performs better in calculating

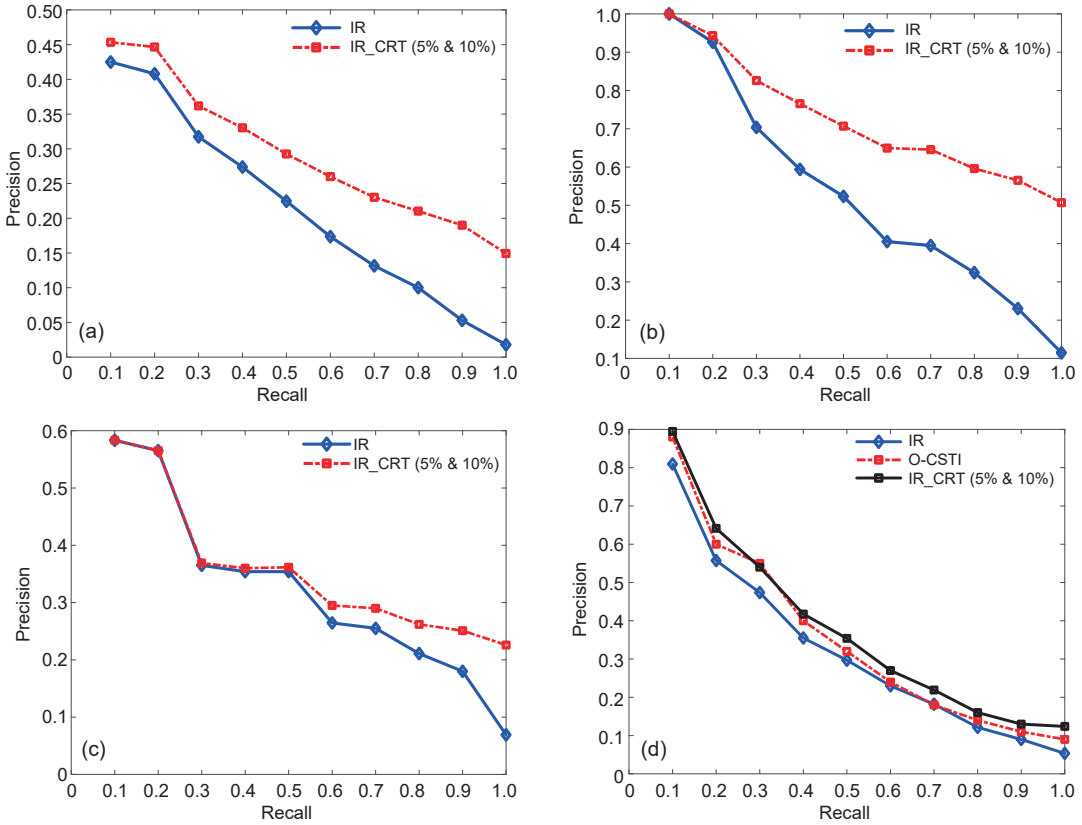

Fig. 8 Comparison of the precision–recall curves of IR versus IR_CRT (5% & 10%): (a) CM1-NASA; (b) Pine; (c) GANNT; (d) iTrust (since the type of its target artifact is code, we also add the O-CSTI result for comparison)

the close relations between textual documents in a natural language. For source code, many programming language reserved words do little to calculate close relations between code. There are also some abbreviations and custom variable names in code that need to be pre-processed. Moreover, the structural information between the code is richer. When the source code is treated only as a natural language text, part of the semantic information of the source code is lost. On the contrary, for requirement documents and test cases, most of which are made up of natural languages, the word embedding results will be better, for the final precision and recall values.

As the results show, the IR_CRT method can effectively improve the performance of the IR model. As an example of the source artifact UC_7 in the EasyClinic dataset, Table 2 describes the top 10 target artifacts' re-ranking process. According to the combination of cut percentages 5% and 10%, the initial requirement region is $\{TC_{51}, TC_{56}\}$, CRTG=$\{\{TC_{51}, TC_{53}, TC_{56}, TC_{55}, TC_{52}, TC_{54}\}, \{TC_{56}, TC_{57}, TC_{51}, TC_{53}, TC_{55}, TC_{54}\}\}$, namely, E=$\{(TC_{51}, TC_{51}), (TC_{51}, TC_{53}), (TC_{51}, TC_{56}), (TC_{51}, TC_{55}), (TC_{51}, TC_{52}), (TC_{51}, TC_{54}), (TC_{56}, TC_{56}), (TC_{56}, TC_{57}), (TC_{56}, TC_{51}), (TC_{56}, TC_{53}), (TC_{56}, TC_{55}), (TC_{56}, TC_{54})\}$. The results show that the trace links, which are ranked 8^{th} and 9^{th}, rise to 6^{th} and 7^{th} after re-ranking, respectively.

6 Discussion

In the RT, IR is often used to automate the creation of trace links. This can reduce a lot of labor and time. However, the results are often not so satisfactory, so there is a lot of research on improving IR. In this paper, we improve the preliminary ranking lists of IR by leveraging close relations between the target artifacts, and the experimental results suggest that this method can effectively improve the IR performance from the perspective of the three metrics (precision, recall, and MAP).

Table 2 An example of the re-ranking part of the target artifacts of UC_7 (EasyClinic)

Source artifact	Target artifact	Similarity score (initial)	Ranking (initial)	Similarity score (recalculated)	Re-ranking	isTrace
UC_7	TC_{51}	0.320 63	1	0.423 433 596 9	1	×
UC_7	TC_{56}	0.289 95	2	0.382 916 668 5	2	×
UC_7	TC_{55}	0.286 59	3	0.378 479 351 7	3	×
UC_7	TC_{54}	0.282 88	4	0.373 579 814 4	4	×
UC_7	TC_{52}	0.279 67	5	0.369 340 592 1	5	×
UC_7	TC_{58}	0.273 86	6	0.273 86	8↓	
UC_7	TC_{62}	0.273 27	7	0.273 27	9↓	
UC_7	TC_{53}	0.271 60	8	0.358 683 108	6↑	×
UC_7	TC_{57}	0.232 51	9	0.307 059 681 3	7↑	×
UC_7	TC_{60}	0.041 22	10	0.053 586	10	

UC: use case; TC: test case; ×: trace link

6.1 Related work

To improve the accuracy of the IR-based traceability recovery method, the current main idea is from the perspective of lexical analysis, such as text pre-processing (Zhao et al., 2017) and the IR model (Ghannem et al., 2017). Despite these advances, no single IR method has been shown to consistently have the highest performance for traceability (Abadi et al., 2008; Oliveto et al., 2010; Gethers et al., 2011).

In addition to these technical improvements, other related work has concentrated on the characteristics of the target artifacts in traceability. The most studied is that when the target artifact is code, the dependency between code should be considered to improve the performance of the IR model. Panichella et al. (2013) proposed an algorithm, optimistic combination of structural and textual information, called O-CSTI, which uses an IR method to locate a set of initial trace links, and then extends that set using the structural information. The similarity in each link is increased by adding a bonus δ if there is a relation between the class in the current link and other classes. Based on O-CSTI, the method UD-CSTI proposed by Panichella et al. (2013) introduces user feedback and code dependency analysis into IR-based requirement-to-code traceability recovery. Although these methods can improve accuracy, they require extra manpower and degrade the automation of the tool.

Based on the related work, we have proposed a new method (called IR_CRT) combining IR techniques and close relations between target artifacts, which considers any type of target artifact and thus has a kind of universality. In other words, our method is more applicable. The target artifact of RT includes, but is not limited to, source code; it can also be use cases, test cases, design, and so on. Moreover, the method is fully automated and requires no human intervention. Experimental results showed that our method is superior to the baseline method.

To support traceability, several tools have been introduced, including ADAMS Re-Trace (de Lucia et al., 2005), RETRO (Dekhtyar et al., 2007; Hayes et al., 2007; Hayes, 2012), Poirot (Lin et al., 2006), Tracter (Mahmoud and Niu, 2011), and TraceLab (Keenan et al., 2012). Each of these tools has a substantial code base, test cases, and sometimes user stories or requirements, design, etc. For ease of use and accessibility, we experiment with RETRO, which provides the original list of candidates for the first step of the experiment.

Recently, there has also been some seminal work on traceability. For example, Guo et al. (2017) introduced a method which uses deep learning to incorporate requirement artifact semantics and domain knowledge into the tracing solution. Unfortunately, because of the limitations of the obtained datasets (difficult to acquire large corpora), such methods cannot be applied in practice. Du et al. (2020) proposed an active learning method to train a predictive tracing link model with less information, which greatly reduced the manpower.

6.2 Threats to validity

Some threats could potentially limit the validity of our experiments.

First, a possible threat to the validity of this method is the quality and size of the initial

requirement region. Low quality of artifacts and unsuitable IR techniques could negatively affect the ranking of the candidate list. In addition, the strategy (cut-point based or threshold based) for cutting the list has an impact. Different values for cut-point or threshold will lead to different sizes of the initial requirement region. In view of this threat, some low-quality artifacts can be enhanced by using thesaurus, query extension, the Rocchio algorithm, and other techniques. On the basis of good quality of the artifacts, different IR techniques such as VSM, LSI, and PM can be applied to carry out experiments. When selecting a cut percentage to form the initial requirement region, experiments were carried out on multiple datasets, and the current optimum was selected for subsequent experiments.

Second, a possible threat to the validity of our method is the quality of close relations ranking between target artifacts. The inclusion of irrelevant texts in artifacts in pre-trained word embedding models can lead to inaccurate and incomplete ranking results. To cut the irrelevant text in the artifacts, we need to take more pre-processing to solve the problems of synonyms, polysemy, and reserved words. There is still room for improvement in the accuracy of the embedded model proposed in this paper. Because of the limited datasets in hand at present, a general domain model was temporarily used to calculate the close relations. In future, we will use more datasets to train specific domain models to improve the quality of close relations ranking between target artifacts.

Third, a possible threat to the validity of our method is the selection of evaluated systems. We evaluated several medium-sized systems. However, it is hard to estimate the performance of our method for large systems. The method has not yet been validated on large datasets because the datasets available so far are of medium size. Some large datasets, such as industrial ones, are temporarily unavailable. In future work, we will use some large systems to verify this method.

7 Conclusions

For requirement traceability recovery, pure IR-based methods (which measure only the textual similarity of two artifacts) have hit a bottleneck. In this paper, we have proposed a new method, IR_CRT, to generate trace links automatically via combining IR techniques and close relations between target artifacts. Besides textual similarity, the close semantic relations between target artifacts have been considered. An empirical evaluation conducted on five software systems suggests that our approach outperforms the baseline method. The advantages of this method are that it can effectively improve the performance of the IR model without considering the type of the target artifact (not limited by the datasets) and without the need for additional labor (fully automated).

In future work, we will generate trace links across artifacts using close relations between target artifacts, not just target artifacts of the same type. At the same time, we will apply this method to more datasets of different sizes to prove the effectiveness of the method.

Contributors

Haijuan WANG designed the research, processed the data, and drafted the manuscript. Guohua SHEN and Zhiqiu HUANG helped organize the manuscript. Yaoshen YU and Kai CHEN revised and finalized the paper.

Compliance with ethics guidelines

Haijuan WANG, Guohua SHEN, Zhiqiu HUANG, Yaoshen YU, and Kai CHEN declare that they have no conflict of interest.

References

Abadi A, Nisenson M, Simionovici Y, 2008. A traceability technique for specifications. Proc 16th IEEE Int Conf on Program Comprehension, p.103-112. https://doi.org/10.1109/ICPC.2008.30

Antoniol G, Canfora G, Casazza G, et al., 2002. Recovering traceability links between code and documentation. *IEEE Trans Softw Eng*, 28(10):970-983. https://doi.org/10.1109/TSE.2002.1041053

Armstrong BT, 2013. Can Clustering Improve Requirements Traceability? A Tracelab-Enabled Study. MS Thesis, California Polytechnic State University, San Luis Obispo, USA. https://doi.org/10.15368/theses.2013.231

Burgstaller B, Egyed A, 2010. Understanding where requirements are implemented. Proc IEEE Int Conf on Software Maintenance, p.1-5. https://doi.org/10.1109/ICSM.2010.5609699

Chen XF, Grundy J, 2011. Improving automated documentation to code traceability by combining retrieval techniques. Proc 26th IEEE/ACM Int Conf on Automated Software Engineering, p.223-232. https://doi.org/10.1109/ASE.2011.6100057

Dekhtyar A, Hayes JH, Larsen J, 2007. Make the most of your time: how should the analyst work with automated

traceability tools? Proc 3rd Int Workshop on Predictor Models in Software Engineering, Article 4. https://doi.org/10.1109/PROMISE.2007.8

de Lucia A, Fasano F, Oliveto R, et al., 2005. ADAMS Re-Trace: a traceability recovery tool. Proc 9th European Conf on Software Maintenance and Reengineering, p.32-41. https://doi.org/10.1109/CSMR.2005.7

Dietrich T, Cleland-Huang J, Shin Y, 2013. Learning effective query transformations for enhanced requirements trace retrieval. Proc 28th IEEE/ACM Int Conf on Automated Software Engineering, p.586-591. https://doi.org/10.1109/ASE.2013.6693117

Du TB, Shen GH, Huang ZQ, et al., 2020. Automatic traceability link recovery via active learning. *Front Inform Technol Electron Eng*, 21(8):1217-1225. https://doi.org/10.1631/FITEE.1900222

Gethers M, Oliveto R, Poshyvanyk D, et al., 2011. On integrating orthogonal information retrieval methods to improve traceability recovery. Proc 27th IEEE Int Conf on Software Maintenance, p.133-142. https://doi.org/10.1109/ICSM.2011.6080780

Ghannem A, Hamdi MS, Kessentini M, et al., 2017. Search-based requirements traceability recovery: a multi-objective approach. Proc IEEE Congress on Evolutionary Computation, p.1183-1190. https://doi.org/10.1109/CEC.2017.7969440

Gotel OCZ, Finkelstein CW, 1994. An analysis of the requirements traceability problem. Proc IEEE Int Conf on Requirements Engineering, p.94-101. https://doi.org/10.1109/ICRE.1994.292398

Guo J, Cheng JH, Cleland-Huang J, 2017. Semantically enhanced software traceability using deep learning techniques. Proc IEEE/ACM 39th Int Conf on Software Engineering, p.3-14. https://doi.org/10.1109/ICSE.2017.9

Hayes JH, 2012. Be the change you want to see in requirements engineering: a letter to myself. Dagstuhl Seminar, Requirements Management-Novel Perspectives and Challenges (12442).

Hayes JH, Dekhtyar A, Sundaram SK, 2005. Improving after-the-fact tracing and mapping: supporting software quality predictions. *IEEE Softw*, 22(6):30-37. https://doi.org/10.1109/MS.2005.156

Hayes JH, Dekhtyar A, Sundaram SK, 2006. Advancing candidate link generation for requirements tracing: the study of methods. *IEEE Trans Softw Eng*, 32(1):4-19. https://doi.org/10.1109/TSE.2006.3

Hayes JH, Dekhtyar A, Sundaram SK, et al., 2007. REquirements TRacing On target (RETRO): improving software maintenance through traceability recovery. *Innov Syst Softw Eng*, 3(3):193-202. https://doi.org/10.1007/s11334-007-0024-1

Keenan E, Czauderna A, Leach G, et al., 2012. TraceLab: an experimental workbench for equipping researchers to innovate, synthesize, and comparatively evaluate traceability solutions. Proc 34th Int Conf on Software Engineering, p.1375-1378. https://doi.org/10.1109/ICSE.2012.6227244

Kuang HY, Nie J, Hu H, et al., 2017. Analyzing closeness of code dependencies for improving IR-based traceability recovery. Proc IEEE 24th Int Conf on Software Analysis, Evolution and Reengineering, p.68-78. https://doi.org/10.1109/SANER.2017.7884610

Leuser J, Ott D, 2010. Tackling semi-automatic trace recovery for large specifications. Proc 16th Int Working Conf on Requirements Engineering: Foundation for Software Quality, p.203-217. https://doi.org/10.1007/978-3-642-14192-8_19

Lin J, Lin CC, Cleland-Huang J, et al., 2006. Poirot: a distributed tool supporting enterprise-wide automated traceability. Proc 14th IEEE Int Requirements Engineering Conf, p.363-364. https://doi.org/10.1109/RE.2006.48

Mahmoud A, Niu N, 2011. TraCter: a tool for candidate traceability link clustering. Proc IEEE 19th Int Requirements Engineering Conf, p.335-336. https://doi.org/10.1109/RE.2011.6051663

Mahmoud A, Niu N, 2013. Supporting requirements traceability through refactoring. Proc 21st IEEE Int Requirements Engineering Conf, p.32-41. https://doi.org/10.1109/RE.2013.6636703

Mahmoud A, Niu N, 2014. Supporting requirements to code traceability through refactoring. *Requir Eng*, 19(3):309-329. https://doi.org/10.1007/s00766-013-0197-0

Marcus A, Maletic JI, 2003. Recovering documentation-to-source-code traceability links using latent semantic indexing. Proc 25th Int Conf on Software Engineering, p.125-135. https://doi.org/10.1109/ICSE.2003.1201194

Oliveto R, Gethers M, Poshyvanyk D, et al., 2010. On the equivalence of information retrieval methods for automated traceability link recovery. Proc IEEE 18th Int Conf on Program Comprehension, p.68-71. https://doi.org/10.1109/ICPC.2010.20

Panichella A, McMillan C, Moritz E, et al., 2013. When and how using structural information to improve IR-based traceability recovery. Proc 17th European Conf on Software Maintenance and Reengineering, p.199-208. https://doi.org/10.1109/CSMR.2013.29

Wang XB, Lai GH, Liu C, 2009. Recovering relationships between documentation and source code based on the characteristics of software engineering. *Electron Notes Theor Comput Sci*, 243:121-137. https://doi.org/10.1016/j.entcs.2009.07.009

Zhao T, Cao QH, Sun Q, 2017. An improved approach to traceability recovery based on word embeddings. Proc 24th Asia-Pacific Software Engineering Conf, p.81-89. https://doi.org/10.1109/APSEC.2017.14

Zou XC, Settimi R, Cleland-Huang J, 2006. Phrasing in dynamic requirements trace retrieval. Proc 30th Annual Int Computer Software and Applications Conf, p.265-272. https://doi.org/10.1109/COMPSAC.2006.66

Zou XC, Settimi R, Cleland-Huang J, 2010. Improving automated requirements trace retrieval: a study of term-based enhancement methods. *Empir Softw Eng*, 15(2):119-146. https://doi.org/10.1007/s10664-009-9114-z