IRRT: An Automated Software Requirements Traceability Tool based on Information Retrieval Model

Sen Zhang[1], Hongyan Wan[1,*], Yong Xiao[1], and Ziruo Li[2]
[1]School of Computer Science and Artificial Intelligence, Wuhan Textile University, Wuhan, China
[2]School of Information Engineering, Wuhan University of Technology, Wuhan, China
zhangsen7373@163.com, hywan@wtu.edu.cn, 634218476@qq.com, ziruo_l@whut.edu.cn
*corresponding author

***Abstract*—In the software development process, requirements traceability is a key part for ensuring the success of the entire project. It is very important to generate requirements traceability links, which can promote the software development and maintenance processes, such as software requirements integrity analysis, software evaluation, software testing, software validating, etc. However, the generation of the requirements traceability links is usually time-consuming and labor-intensive. In order to solve it, we designed and developed an automated software requirements traceability tool based on information retrieval (IR) model. The tool can not only automatically generate trace links but also evaluate trace links. It uses Vector Space Model (VSM) and the trace recommendation-based code class structure (TRCCS) links to generate trace links. We measure the performance in term of Precision, Recall, and F_2 are used to evaluate the trace links. The experimental results show that the tool can improve the efficiency of requirements traceability links generation and better support software development activities.**

Keywords-requirements traceability tool; information retrieval; vector space model; IRRT

I. INTRODUCTION

In the software development process, some requirements are often omitted or unrealized, and the impact of requirements modifications on software development is regularly ignored. These small issues may lead to the failure of the whole software development [1,2]. Requirements traceability can act tracing analysis to decide whether or not all low-level elements have parent elements. Furthermore, completeness analysis is carried out to determine whether all high-level requirements are absolutely satisfied, and test coverage is assessed to determine whether test cases exist for each requirement [3,4,5,6]. Therefore, requirements traceability is an essential research direction in requirements engineering, and has been the topic of active research by academia and industry.

Manual generation of requirements traceability links has many limitations. It requires a giant quantity of professionals who are acquainted with the project, and a lot of time when dealing with complicated software system. In addition, during the actual software development process, the modification in the requirements is inevitable. However, it is difficult to maintain the update the trace links, and it is useless to not save the latest links in time. Thus, it is meaningful to design and develop a tool that can generate requirements traceability links automatically. In this paper, we designed and developed an IR-based requirements traceability (IRRT) tool, which can automate the preprocessing of software artifacts ((e.g., requirements documents, source code) and generate links between them, with the ability to modify trace links in real time. The problem of time-consuming and labor-intensive generation of requirements traceability links and untimely link updates can be well solved with the tool. It also can reduce the problem of multiple synonyms.

The IRRT tool realizes the generation of trace links through the commonly used IR model Vector Space Model (VSM) [7,8,9], and refines the links through the trace recommendation-based code class structure (TRCCS). The principal functions are reflected in the following three aspects.

- IRRT realizes the automatic generation of requirements traceability links by using VSM.
- IRRT refines candidate links by adding TRCCS, and significantly improves the precision while keeping the general effect of trace links unchanged or improving slightly.
- IRRT evaluates the generated trace links through three metrics: Precision, Recall, and F_2, and displays them in the form of a line chart.

The remainder of the paper is organized as follows: Section II introduces the background of requirements traceability and IR based requirements traceability, as well as some commonly used IR models. Section III introduces the tool IRRT and the specific functions of the tool in detail. Section IV validates the tool through a specific case study. Section V discuss our results and related work. Section VI summarizes this study and the outlook for future development.

II. BACKGROUND

Requirements traceability is a critical part of requirements engineering. For decades, many requirements traceability technologies have been widely studied [10]. Requirements traceability is to describe and trace the lifecycle of requirements, both in a forward and backward direction [11]. Requirements traceability generates links between requirements documents and other software artifacts, including requirements documents, source code, use cases, test cases, design documents, and so on. It plays a significant role in the software development process. The requirements which defined in the early stage of software development are

2693-9371/22/$31.00 ©2022 IEEE
DOI 10.1109/QRS-C57518.2022.00084

frequently exchanged during the software lifecycle. It is crucial to analyze the effects of requirement adjustments. Requirements traceability links can be used to determine which software artifacts need to be adjusted when requirements change.

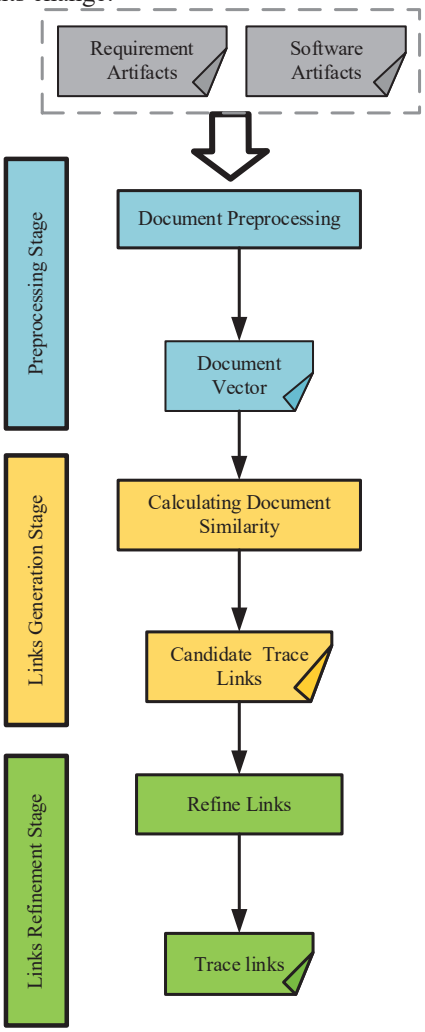

Figure 1. The general process of IR-based trace links generation

At present, there are various technologies used to generate requirements traceability links, including IR-based technology, machine learning-based technology, data mining-based technology, etc. among which the IR-based technique is most commonly used. As shown in Figure 1, the general process of IR-based method is divided into three stages: **First: Preprocessing Stage,** which converts requirements documents, source code, test cases and other software artifacts into vector form. **Second: Links Generation Stage**, the relevance of two artefacts is determined by calculating their similarity, and then the candidate trace links list is created to generate links. **Third: Links Refinement Stage,** manual selection or some other techniques are used to refine the candidate trace links, producing higher quality trace links.

Representing documents as vectors in a vector space is called VSM [12]. VSM represents documents as vectors by terms and their corresponding weights, where term frequency-inverse document frequency (TF-IDF) [13] is used to calculate the weights of the terms. The weight of the terms t_i corresponding to w_i is calculated by the following equation:

$$w_i = tf(t_i) \times idf_i \tag{1}$$

VSM calculates the similarity of vectors to determine how similar the documents are. Typical techniques for determining document similarity include: Euclidean Distance, Manhattan Distance and Cosine Distance, etc. Cosine distance is the most frequently used, and VSM employs it to assess how similar two documents are. The cosine distance is calculated using the following equation:

$$sim(q_i, d_i) = \frac{\sum_1^N (w_i, q_i)}{\sqrt{\sum_1^N w_i^2} \times \sqrt{\sum_1^N q_i^2}} \tag{2}$$

By comparing the cosine distance between the documents, it is possible to determine how relevant they are and generate a trace link. The generation of requirements traceability links uses a variety of IR models, the most popular of which are the VSM and Jensen-Shannon (JS) models [14]. The tool uses VSM to generate trace links and further refines the generated links through TRCCS.

III. INTRODUCTION OF TOOLS

A. Overview

The major functions of IRRT include four parts: generating trace links, refining trace links, evaluating trace links and building dictionaries. The generation of the trace links is divided into manual generation and automatic generation. VSM is frequently employed in requirements traceability field, is used for the automatic generation of trace links. For the purpose of improving the trace links' quality, TRCCS is used to further refine the links that VSM generated. The evaluation of trace links uses Precision, Recall, and F_2, which are the most commonly used evaluation metrics in the requirements traceability field, and the statistical graphs of results are displayed in the tool. By building a dictionary, the problem of multiple synonyms that frequently occurs in generating trace links is alleviated. We use a dictionary to solve the problem of being unable to generate trace links when two terms are semantically similar or identical. The main interface of IRRT is shown in Figure 2, including six parts: File, Manual Trace, Auto Trace, Evaluation, Tools and Help. They are described in detail below.

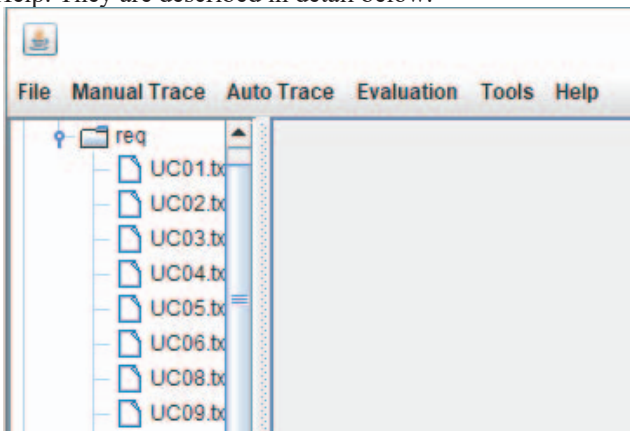

Figure 2. The IRRT interface

B. Create project

The first step of IRRT is to create a new project. As shown in Figure 3, it includes four project documents: test cases, requirements documents, design documents, and source code. After importing the documents, their details can be shown from the tool.

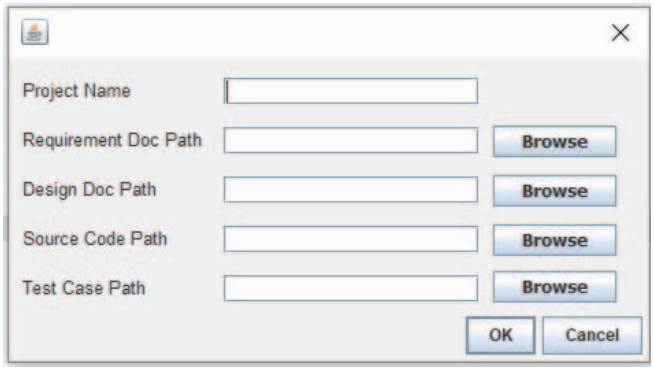

Figure 3. Create project interface of IRRT

C. Manual Trace

IRRT supports manual generation of the trace links. The user can select two documents and determine whether a link can be generated. This function is necessary. For small projects, it is more accurate and effective to generate trace links manually. This function is also an improvement to the automatic generation of the trace links. Figure 4 shows the manual trace interface of IRRT. As shown in Figure 4, we can choose two documents, click Build to create a link, click Save to save it, and click Del to remove any unneeded or incorrect links.

D. Auto Trace

As shown in Figure 5, the IRRT generates requirements traceability links in two steps: at first, click VSM to begin the automatic generation of trace links, and then click TRCCS to refine the candidate trace links.

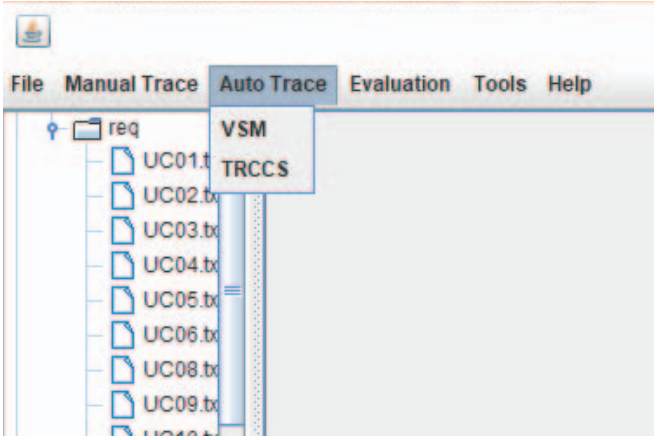

Figure 5. Auto trace interface of IRRT

1) Generate trace links: at first, pre-processing the documents, including word segmentation, stop word removal, part-of-speech tagging and stemming, etc. Then, in order to generate trace links, VSM is used to calculate the similarity between documents. The greater the similarity, stronger the relevance. Documents are converted to vectors by VSM, which are made up of terms and their related weights. VSM uses the TF-IDF algorithm to calculate the weights of terms, document D is expressed as $D_i = \{t_i, \ldots, t_n\}$, and its vector form is $V = \{w_1, \ldots, w_n\}$, the weight w_i corresponding to term t_i is calculated by (1). The similarity is calculated by using the

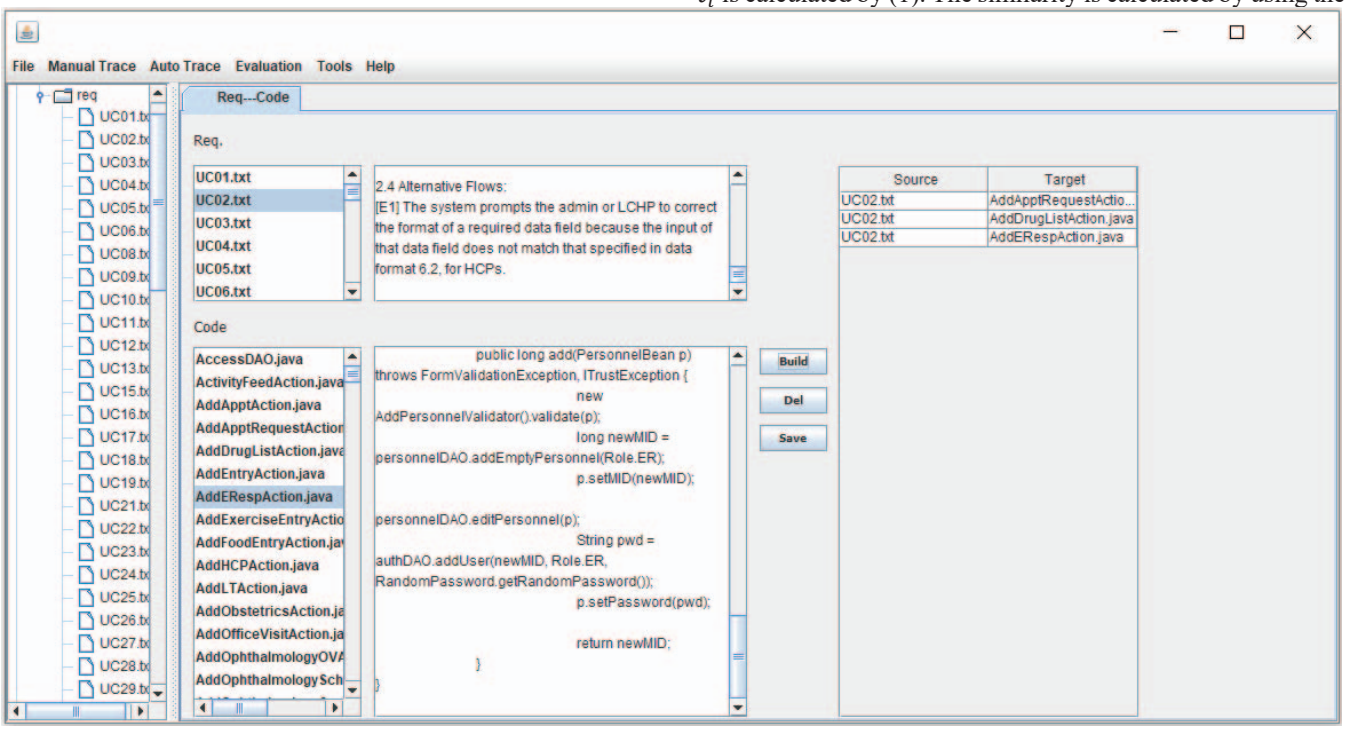

Figure 4. Manual trace interface of IRRT

cosine distance between vectors of two documents. The cosine distance is calculated by (2). The range of similarity values is from 0 to 1. The higher the value, the more relevant the document is, and vice versa.

2) Refine trace links: Refining candidate trace links by TRCCS, which uses the existing code-class structure relationship of trace links to recover more trace links. TRCCS recovers the trace links that cannot be established correctly due to the semantics mismatch, uncertain requirements or annotations. Therefore, the quality of the trace links can be improved. Based on the similarity calculated by VSM, TRCCS calculates the similarity between the code artifacts with the equation (3):

$$TRCCS(q_i, d_i) = sim(q_i, d_i) + \left(1 - sim(q_i, d_i)\right) \times \frac{\sum_1^{|\phi(d_i)|} sim(q_i, d_k)}{|\phi(d_i)|}, d_k \in \phi(d_i) \quad (3)$$

According to the similarity value calculated by using TRCCS, the candidate trace links list is reordered to refine the trace links.

E. Evaluation

To evaluate the quality of the generated trace links, we select three metrics which are commonly used in the requirements traceability field: Precision, Recall, and F_2. Their calculation methods are shown in equation (4)、(5)、(6), respectively.

$$precision = \frac{|\, correct \cap retrieved \,|}{|\, retrieved \,|} \quad (4)$$

$$recall = \frac{|\, correct \cap retrieved \,|}{|\, correct \,|} \quad (5)$$

where *correct* represents the set of correct trace links, *retrieved* represents the set of all generated links, and $\cap$ represents the intersection of two sets. F_2 is the weighted harmonic mean of precision and recall. When $\beta = 2$, the precision weight is small and the recall weight is large [15]. We choose F_2 to evaluate the quality of trace links.

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{\beta^2 \times precision + recall} \quad (6)$$

To evaluate the quality of the created links, click the Evaluation button and then choose the path of the generated links and the path of the true links. The evaluation results are displayed on the tool's main interface, and the Add Data button allows you to add additional results from other models for comparison. Click the Details button to display detailed results, it shows the results of evaluation for each requirement use case and the complete average results. In addition, the threshold can be adjusted by using the Change Threshold button.

F. Tools

VSM is usually based on terms matching to calculate the similarity between requirements documents and other documents. The advantage of VSM is that a straightforward linear algebraic model can be used to implement it. But there is have a problem. When the semantics of two terms are similar while the calculated similarity is low, the two terms cannot be matched, which is inconsistent with the actual situation. The likelihood of the multiple synonyms problem is decreased through manually adding common words with similar semantics to create a dictionary. As shown in Figure 6, click the Tools button and select Thesaurus to enter the dictionary creation interface.

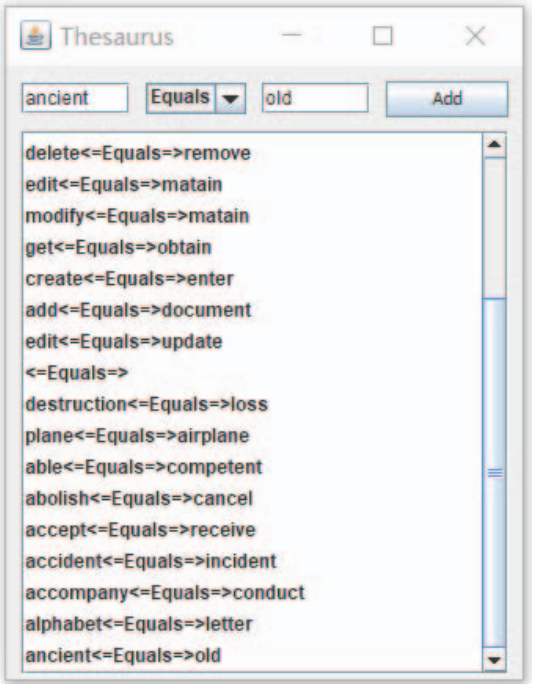

Figure 6. Auto trace interface of IRRT

G. Help

This part is mainly for the detailed introduction of the two technologies VSM and TRCCS used in the tool. Users can learn more about the two technologies to better use the tool by viewing a comprehensive description of them in the tool's primary interface.

IV. CASE STUDY

A. Datasets

IRRT is validated for all datasets on the Center of Excellence for Software and Systems Traceability (CoEST[1]). The open source datasets above CoEST are widely used in the research of requirements traceability [16,17,18,19]. Table 1 shows the types and number of source artifacts and target artifacts for each dataset, along with the number of true links for each dataset. In this paper, we select iTrust [20,21,22] to demonstrate how the tool works specifically. iTrust is a java-based medical record web system that includes 50 requirements use cases and 299 Java classes, with a total of 314 true links. The case study focuses on generating links between the requirements documents and the source code.

[1]http://www.coest.org/

Table 1. Applicable datasets for IRRT

Name	Source Artifacts (Number)	Target Artifacts (Number)	Size	True Links
iTrust	Requirements (50)	Code Classes (299)	14950	314
eTour	Use Cases (63)	Code Classes (101)	6363	365
EasyClinic	Use Cases (30)	Code Classes (47)	1410	93
	UML Interaction Diagram (20)	Code Classes (47)	940	69
		Test Cases (63)	1260	83
		Use Cases (30)	600	26
	Test Cases (63)	Code Classes (47)	2961	204
		Use Cases (30)	1890	63
CM-1 NASA	High-level Requirements (22)	Low-level Requirements (53)	1166	45
EBT	Requirements (40)	Test Cases (25)	1000	51
GANNT	High-level Requirements (17)	Low-level Requirements (69)	1173	68
Albergate	Requirements (55)	Code Classes (17)	935	53
WARC	Non-functional Requirements (21)	Software Requirements Specification (89)	1869	58
	Functional Requirements (43)		3827	78
GANNT	High-level Requirements (17)	Low-level Requirements (69)	1173	68
SMOS	Use Cases (56)	Code Classes (101)	5656	1044
eAnci	Use Cases (140)	Code Classes (55)	7700	554
HIPAA	Requirements (1889)	Regulatory Codes (10)	18890	244
Ice Breaker	Requirements (unclear)	UML classes (unclear)	unclear	unclear
Infusion Pump	Requirements (unclear)	Components (unclear)	unclear	unclear
Kiosk	Requirements (unclear)	Process-Specifications (unclear)	unclear	unclear

B. *Generate Trace Links*

1) Preprocessing: After clicking the VSM in Auto Trace, documents are preprocessed. For source code, the tool uses Javadoc2 technology to extract the annotation information of class and method names, generating help files in HTML format to obtain information strongly relevant to system functions. The Apache OpenNLP[3] toolkit is used to perform preprocessing operations on natural language texts such as requirements documents and design documents.

2) VSM: After preprocessing, the VSM calculates the similarity between the requirements document and the source code to generate trace links. The results of the experiment generation links are shown in Figure 7, with the Score value on the right showing the relevance between the source artifact and the target artifact. As shown in Figure 8, in order to generate better trace links, you can manually modify the generated links tags by clicking the Confirm Tags button.

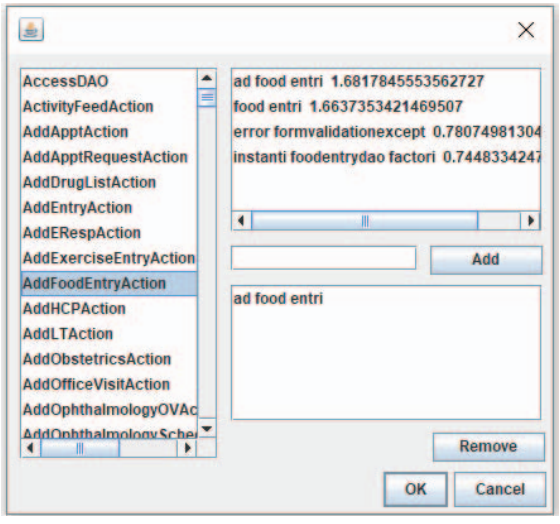

Figure 8. Confirm tags interface of IRRT

[2]http://www.oracle.com/java/technologies/javase/javadoc-tool.ht···

[3]http://opennlp.apache.org/

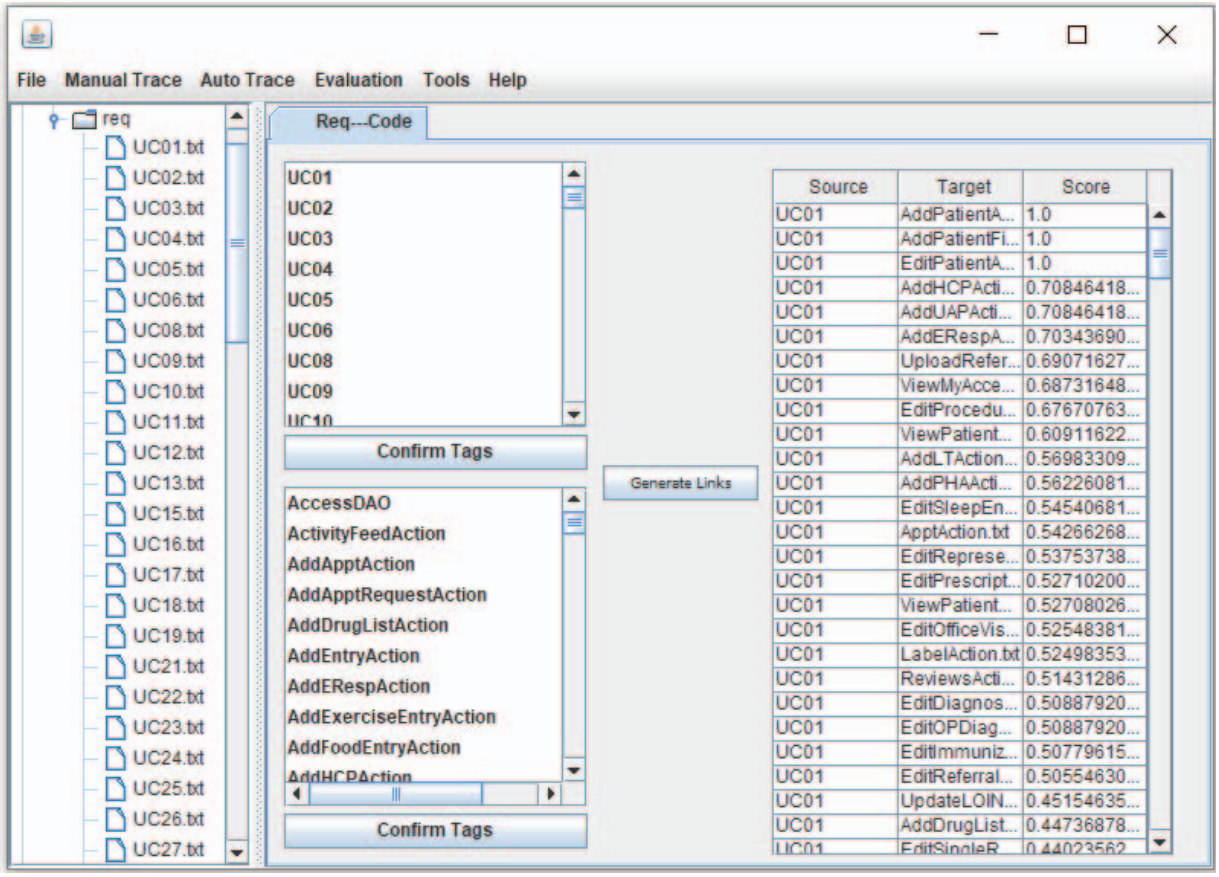

Figure 7. Results for VSM technique

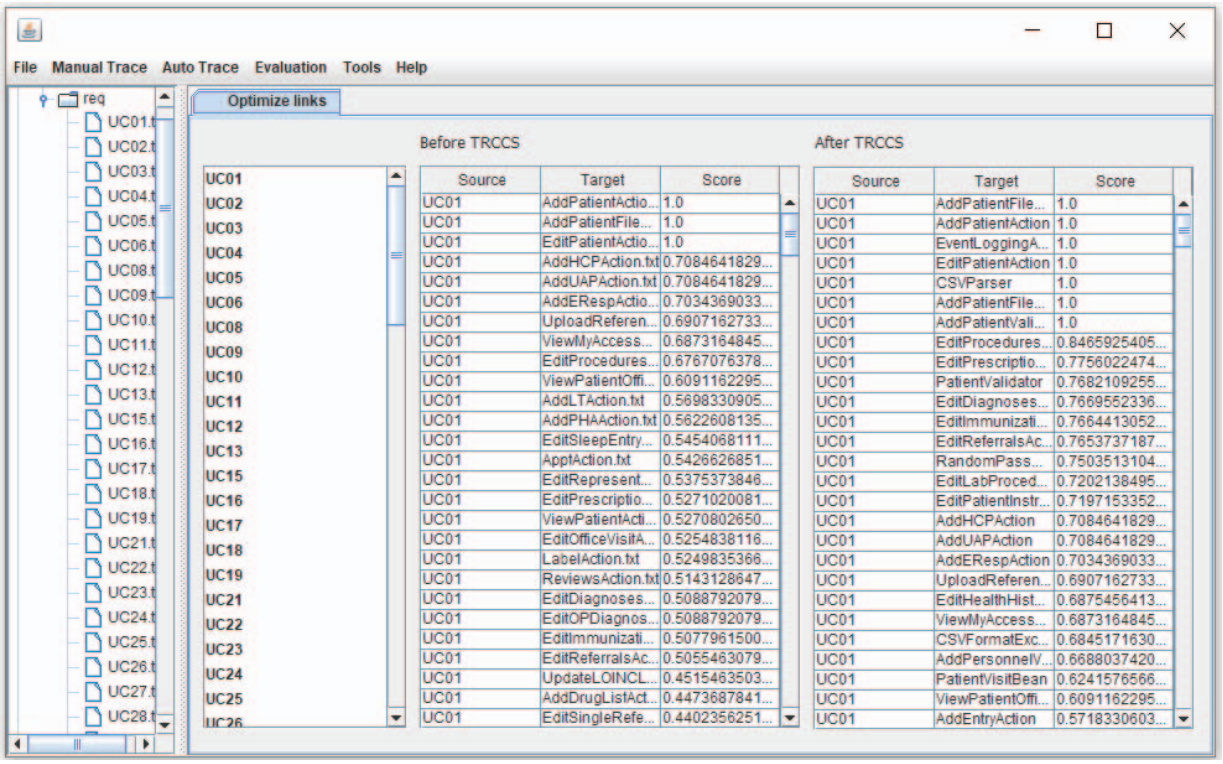

Figure 9. Refine links results via TRCCS

3) TRCCS: In order to recover more trace links, reordering the candidate list of links according to calculating the similarity between code artifacts. The refined results are shown in Figure 9. The left side represents the results before refinement, while the right side is the value after refinement. You can see that a number of trace links are recovered.

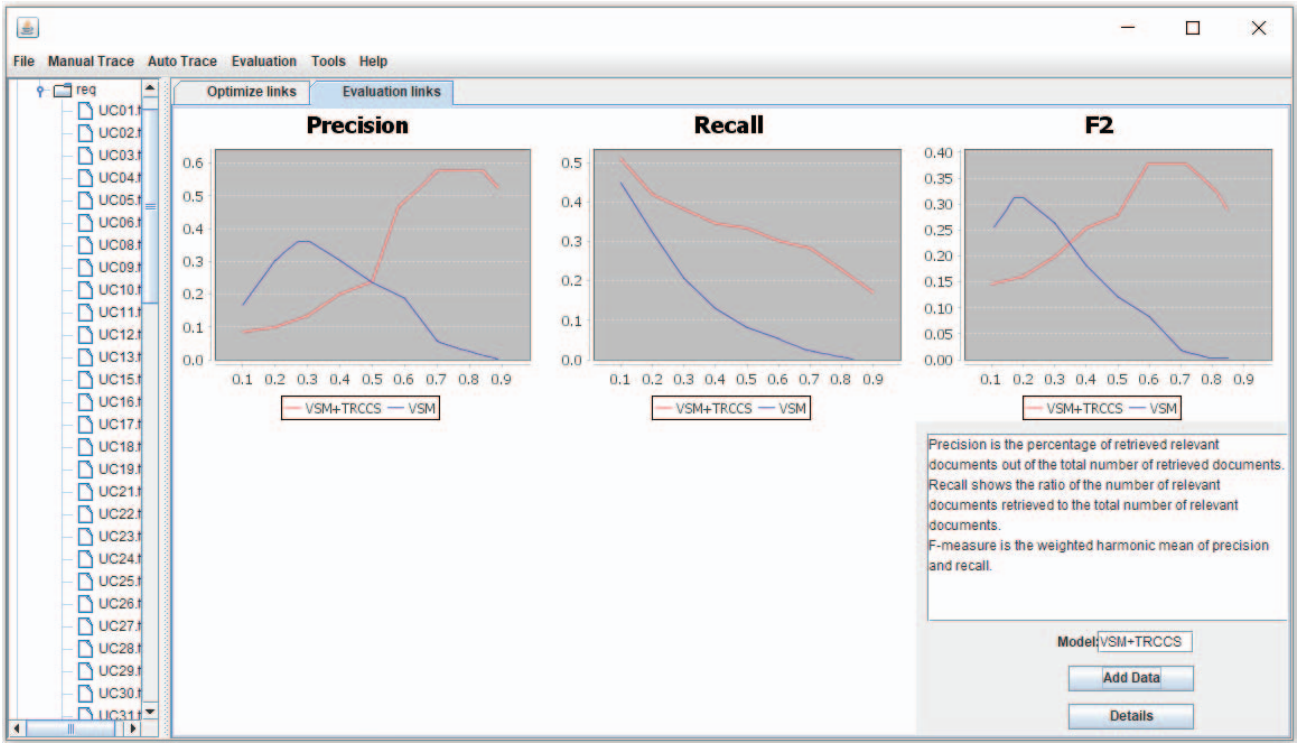

Figure 10. Evaluation results for generated trace links

C. Evaluation Trace Links

We select three metrics: Precision, Recall, and F_2 to evaluate the generated trace links. The results are shown by line graphs. Figure 10 displays the evaluation results. In figure 10, the horizontal axis represents the threshold, and the vertical axis represents the value of the three corresponding metrics. The highest precision is 0.58, and the highest recall is 0.53. It can be seen that recall falls as precision rises. When the threshold is 0.7, F_2 reaches its highest value of 0.37. In comparison to VSM, TRCCS has higher precision and slightly higher recall, because it uses relationships in the code class structures to recover more trace links. The improvement in F_2 is limited because the improvement in recall is not significant. In general, IRRT achieves good effects in generating trace links between requirements and source code.

V. Discussion

Some scholars have also done some work in the field of developing automatic requirements traceability tool. Yoshino et al. [23] proposed a tool for automatically generating initial sequence diagrams from activity diagrams to establish trace links between model elements. The tool uses UML diagrams for requirements analysis, but it is not validated by case study. In contrast, this paper uses document format for requirements analysis and is validated with a case study.

Garcia et al. [24] proposed a prototype of a tool that requires an analyst to manually create a trace link between functional requirements and the implemented functionality of a website. Our tool automatically generate links between requirements-code, requirements-design and requirements-test.

The results of the case study demonstrate that IRRT facilitates the automatic generation of requirements traceability links, but there have two drawbacks.

First, it only uses one model VSM to generate links. For example, when dealing with some documents that focus on semantics, the VSM model does not work well. The VSM performs poorly when utilized to documents that emphasize semantics.

Second, the trace links generated by the numerical display are not intuitive, and the visualization of the trace links is not realized.

VI. Conclusion

In this paper, we developed a tool IRRT to generate the requirements traceability links by using VSM. Specifically, it uses TRCCS to refine the candidate trace links generated by VSM. The automatic generation of the links between requirements-code, requirements-design and requirements-test is realized. In addition, IRRT allows to evaluate the quality of trace links, and reduce the multiple synonyms by building a dictionary. Through the case study, we verified and demonstrated the functions. IRRT aids in the generation of requirements trace links, and improves the efficiency of generating links while ensuring better quality.

The work of this paper can further improve as follows: 1) investigate additional IR models, add more methods for IRRT so that it can be applied to a broader set of software systems. 2) research Machine Learning based requirements traceability recovery, added to IRRT.

ACKNOWLEDGMENT

This work is supported by the Opening Foundation of Engineering Research Center of Hubei Province for Clothing Information (No. 2022HBCI05, No. 2022HBCI02), the National Natural Science Foundation of China Project (No. 62102291) and the Young Talents Programme of Scientific Research Program of the Hubei Education Department (Project No. Q20211711).

REFERENCES

[1] Jarke, Matthias. "Requirements tracing." Communications of the ACM 41.12 (1998): 32-36.

[2] Maza, Sofiane, and Oussama Megouas. “Framework for trustworthiness in software development.” International Journal of Performability Engineering 17.2 (2021): 241–252.

[3] Hayes, Jane Huffman, Alex Dekhtyar, and Senthil Karthikeyan Sundaram. "Improving after-the-fact tracing and mapping: Supporting software quality predictions." IEEE software 22.6 (2005): 30-37.

[4] Li, Dongcheng, W. Eric Wong, et al. “Automatic test case generation using many-objective search and principal component analysis.” IEEE Access 10 (2022): 85518–85529.

[5] Lee, Chen-Hua, and Chin-Yu Huang. “Applying cluster-based approach to improve the effectiveness of test suite reduction,” International Journal of Performability Engineering 18.1 (2022): 1–10.

[6] Li, Dongcheng, W. Eric Wong, et al. “Improving search-based test case generation with local search using adaptive simulated annealing and dynamic symbolic execution.” 2022 9th International Conference on Dependable Systems and Their Applications (DSA). IEEE, 2022.

[7] Rodriguez, Danissa V., and Doris L. Carver. "Multi-objective information retrieval-based NSGA-II optimization for requirements traceability recovery." 2020 IEEE International Conference on Electro Information Technology (EIT). IEEE, 2020.

[8] Wang, Bangchao, et al. "An Automated Hybrid Approach for Generating Requirements Trace Links." International Journal of Software Engineering and Knowledge Engineering 30.07 (2020): 1005-1048.

[9] Rodriguez, Danissa V., and Doris L. Carver. "An IR-based artificial bee colony approach for traceability link recovery." 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2020.

[10] Wang, Bangchao, et al. "Requirements traceability technologies and technology transfer decision support: A systematic review." Journal of Systems and Software 146 (2018): 59-79.

[11] Gotel, Orlena CZ, and C. W. Finkelstein. "An analysis of the requirements traceability problem." Proceedings of ieee international conference on requirements engineering. IEEE, 1994.

[12] Schütze, Hinrich, Christopher D. Manning, and Prabhakar Raghavan. Introduction to information retrieval. Vol. 39. Cambridge: Cambridge University Press, 2008.

[13] Ponte, Jay M., and W. Bruce Croft. "A language modeling approach to information retrieval." ACM SIGIR Forum. Vol. 51. No. 2. New York, NY, USA: ACM, 2017.

[14] Wang, Bangchao, et al. "A Systematic Mapping Study of Information Retrieval Approaches Applied to Requirements Trace Recovery." SEKE. 2022.

[15] Hayes, Jane Huffman, Alex Dekhtyar, and Senthil Karthikeyan Sundaram. "Advancing candidate link generation for requirements tracing: The study of methods." IEEE Transactions on Software Engineering 32.1 (2006): 4-19.

[16] Kuang, Hongyu, et al. "Using frugal user feedback with closeness analysis on code to improve IR-based traceability recovery." 2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC). IEEE, 2019.

[17] Ali, Nasir, et al. "Exploiting Parts-of-Speech for effective automated requirements traceability." Information and Software Technology 106 (2019): 126-141.

[18] Li, Tong, et al. "Combining machine learning and logical reasoning to improve requirements traceability recovery." Applied Sciences 10.20 (2020): 7253.

[19] Chen, Lei, et al. "Enhancing unsupervised requirements traceability with sequential semantics." 2019 26th Asia-Pacific Software Engineering Conference (APSEC). IEEE, 2019.

[20] Hammoudi, Mouna, et al. "On the effect of incompleteness to check requirement-to-method traces." Proceedings of the 36th Annual ACM Symposium on Applied Computing. 2021.

[21] Wang, Wentao, et al. "Complementarity in requirements tracing." IEEE Transactions on Cybernetics 50.4 (2019): 1395-1404.

[22] Wang, Haijuan, et al. "Analyzing close relations between target artifacts for improving IR-based requirement traceability recovery." Frontiers of Information Technology & Electronic Engineering 22.7 (2021): 957-968.

[23] Yoshino, Kaito, and Saeko Matsuura. "Requirements Traceability Management Support Tool for UML Models." Proceedings of the 2020 9th International Conference on Software and Computer Applications. 2020.

[24] Garcia, Jorge Esparteiro, and Ana CR Paiva. "A Requirements-to-Implementation Mapping Tool for Requirements Traceability." J. Softw. 11.2 (2016): 193-200.