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

Automatic traceability link recovery via active learning[*]

Tian-bao DU[†1], Guo-hua SHEN[†‡1,2,3], Zhi-qiu HUANG[†1,2,3], Yao-shen YU[1], De-xiang WU[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 210093, China*

[3]*Key Laboratory of Safety-Critical Software, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China*

[†]E-mail: tbdu_312@outlook.com; ghshen@nuaa.edu.cn; zqhuang@nuaa.edu.cn

Received May 2, 2019; Revision accepted Aug. 12, 2019; Crosschecked May 18, 2020; Published online July 4, 2020

Abstract: Traceability link recovery (TLR) is an important and costly software task that requires humans establish relationships between source and target artifact sets within the same project. Previous research has proposed to establish traceability links by machine learning approaches. However, current machine learning approaches cannot be well applied to projects without traceability information (links), because training an effective predictive model requires humans label too many traceability links. To save manpower, we propose a new TLR approach based on active learning (AL), which is called the AL-based approach. We evaluate the AL-based approach on seven commonly used traceability datasets and compare it with an information retrieval based approach and a state-of-the-art machine learning approach. The results indicate that the AL-based approach outperforms the other two approaches in terms of F-score.

Key words: Automatic; Traceability link recovery; Manpower; Active learning
https://doi.org/10.1631/FITEE.1900222 **CLC number:** TP311

1 Introduction

Traceability link recovery (TLR) refers to establishing traceability relationships between a source artifact set S (e.g., requirement documents) and a target artifact set T (e.g., source codes and test cases) within the same project. The resulting software traceability supports activities including impact analysis, test coverage verification, and requirement coverage verification (Rempel and Mäder, 2017). TLR usually requires humans identify the valid (i.e., two artifacts are related) traceability links in $S \times T$ possible traceability links. There are a large number of possible traceability links in a large software system, so it is tough to manually identify valid traceability links. To address this challenge, many research teams have been working to automate the traceability creation process (Asuncion et al., 2010; Borg et al., 2013). Information retrieval (IR) is a common technology to automate TLR (Panichella et al., 2013). IR-based approaches can make it easier to identify traceability links by recommending the links that are most likely to be the traceability links at the top. In recent years, some research teams have begun to apply machine learning to TLR. Mills and Haiduc (2017b) proposed a TLR approach based on traditional supervised learning (hereafter called the TSL-based approach), which achieved good results. The TSL-based approach uses existing traceability information to train a classifier and then uses this classifier to classify possible traceability links

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 61772270), the National Key Research and Development Project of China (Nos. 2016YFB1000802 and 2018YFB1003902), and the Funding of the Key Laboratory of Safety-Critical Software, China (No. 1015-XCA1816403)

ORCID: Guo-hua SHEN, https://orcid.org/0000-0003-2182-0019 as valid or invalid (i.e., two artifacts are unrelated). Although the accuracy of the TSL-based approach is high, creating traceability information can require a lot of manpower, especially for projects without traceability information. To save manpower, we propose a new TLR approach based on active learning. The main difference between the TSL- and AL-based approaches is that the TSL-based approach randomly selects traceability links for labeling, while the AL-based approach selects traceability links for labeling based on a sample selection strategy. The results show that the AL-based approach outperforms the TSL-based approach when we label the same number of traceability links.

2 Related work

Automatic TLR is a research hotspot in the field of software traceability. Much work has been done to increase the level of automation available for establishing traceability links, most commonly using IR techniques to rank artifacts based on document similarity. IR techniques such as the vector space model (VSM) (Antoniol et al., 2000), latent semantic indexing (LSI) (Marcus et al., 2005), and latent Dirichlet allocation (LDA) (Asuncion et al., 2010) have been applied directly to TLR. The accuracy of IR-based approaches is affected by the problem of "vocabulary mismatch" (e.g., synonymy) (Gethers et al., 2011). To solve this problem, current mainstream research has improved IR-based approaches through lexical analysis such as text preprocessing (Lucia et al., 2012) and the IR model (Marcus and Maletic, 2003). While these improvements achieve certain results, rich and high-quality textual information is required by the target software systems. In practice, however, software systems often lack detailed textual descriptions, which makes the results returned by IR-based approaches not good enough (Cleland-Huang et al., 2005).

In recent years, machine learning has gradually been applied to TLR. Cleland-Huang et al. (2007) first proposed a probabilistic classifier trained on a set of indicator words for non-functional requirements. This was subsequently used for linking regulatory codes with project requirements (Cleland-Huang et al., 2010) and architectural tactics with source codes (Mirakhorli et al., 2012). Li et al. (2015) used a knowledge-rich approach to extend a supervised baseline system with (1) additional training samples derived from annotator rationales and (2) additional features derived from a hand-built ontology. Since IR is affected by query quality, Mills and Haiduc (2017b) and Mills et al. (2018) proposed several features related to query quality. They used query quality features and IR-related features to train a classifier and subsequently used the classifier to identify the validity of each link. For projects without traceability information, all of the above methods require humans create training data (labeled traceability links). To save manpower, we propose an AL-based approach to minimize the amount of training data required to generate an effective predictive model.

3 Active learning based approach

For projects without traceability information, the TSL-based approach requires humans label too many traceability links. To save manpower, we use the AL-based approach to select a small number of representative samples for labeling. The AL-based approach includes the following steps:

1. There are $S \times T$ possible traceability links in a system, and these traceability links form a sample set $D = \{x_1, x_2, \ldots, x_n\}$. The AL-based approach generates a training set including these steps:

(1) Randomly label a small number of samples to initialize a labeled sample set. Let D_{l} denote this labeled sample set and an unlabeled sample set $D_{\mathrm{u}} = D \backslash D_{\mathrm{l}}$.

(2) Train a classifier on D_{l}.

(3) Select an unlabeled sample from D_{u} based on the sample selection strategy and request experts to label the sample.

(4) Add the newly labeled sample to D_{l}.

Repeat steps (2)–(4) until the termination condition is met. Once the termination condition is met, the training set D_{l} is obtained (Section 3.1 introduces the details of active learning).

2. Establish a set of features representing traceability links (Section 3.2 describes features).

3. The number of valid links in the training set is much smaller than that of invalid links. To solve this problem, the training data is balanced by rebalancing techniques (Section 3.3 describes more details about rebalancing techniques).

4. The learning engine trains a classifier C

on the training set, and then C is used to classify unlabeled links. The classification algorithm chosen is random forest (RF), as it is accurate and robust (Breiman, 2001). The results also show that RF classification is the best.

Fig. 1 shows an overview of the AL-based approach. In the following subsections, we will give the details of some steps.

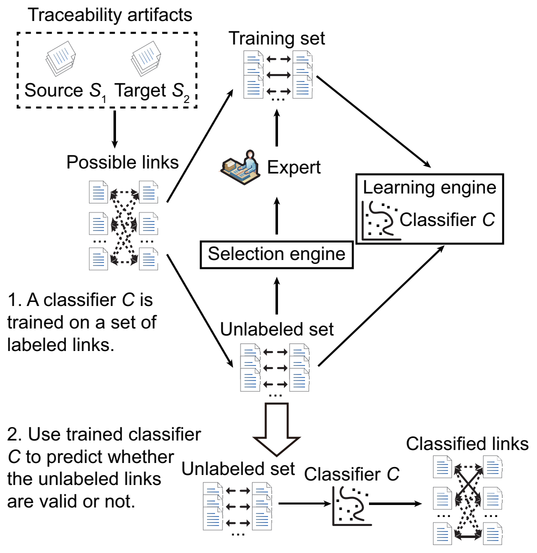

Fig. 1 Overview of the active learning based approach

3.1 Active learning

Active learning allows experts to iteratively label unlabeled samples, and a classifier can be learned or improved at each iteration. An active learning system can generally be divided into two parts: a learning engine and a selection engine. At each iteration, the learning engine trains a classifier on D_l. The selection engine then selects a sample x_i from D_u and requests experts to label x_i before passing it to the learning engine. The learning engine and selection engine alternately work, and the accuracy of the classifier gradually improves. Once the termination condition is met, the iteration will stop. The termination condition we use is that the number of labeled samples reaches a preset value. In our study, the training set size is treated as the preset value. For example, if the training set size is set to N, active learning will stop when the number of labeled samples reaches N. Algorithm 1 gives the active learning process.

Algorithm 1 Active learning process

```
Input: A sample set $D=\{x_1,x_2,\ldots,x_n\}$, a labeled sample
    set $D_\text{l}$, where $D_\text{l}$ is initially empty, and an unlabeled
    sample set $D_\text{u}$, where $D_\text{u}=D\backslash D_\text{l}$
Output: $D_\text{l}$
1: $D_\text{l} \leftarrow D_{\text{l}_0}$  // Randomly label a small number of
                  // samples to initialize $D_\text{l}$
2: Train a classifier on $D_\text{l}$
3: while Termination condition is not met
4:     Select a sample $x_i$ from $D_\text{u}$
5:     Experts label the sample $x_i$
6:     Add the labeled sample $x_i$ to $D_\text{l}$
7:     Train the classifier on $D_\text{l}$
8: end while
9: Return $D_\text{l}$
```

At present, active learning is roughly divided into two categories: stream-based learning and pool-based learning. In stream-based learning, a learning engine receives one sample at a time and has to determine whether to label this sample based on a threshold or not. In general, the threshold of a project is difficult to determine. Thus, active learning primarily focuses on pool-based learning at present, where each query selection is made via search in a fixed unlabeled sample pool. Pool-based sampling selects samples by measuring the uncertainty of samples. A higher uncertainty of the sample leads to higher representativeness and greater likelihood of being selected. The key to pool-based learning is to build a fixed unlabeled sample pool (Cheng et al., 2013). For TLR, unlabeled traceability links can form such a pool. Therefore, pool-based learning is more suitable for TLR.

3.2 Features representing the links

We use two types of features: IR-based features and query quality (QQ) features. IR can capture the textual similarity between software artifacts and build a list of candidate links. This list contains all possible pairs of source and target artifacts ranked according to textual similarities. The ranking of the links in the list of candidate links reflects its validity to some extent. Therefore, our first feature set is about the ranking of links. Since the accuracy of IR is affected by QQ, our second feature set is about QQ.

3.2.1 IR-based features

Given two artifact sets S and T, these is a possible link between artifacts d_1 and d_2, where $d_1 \in S$ and $d_2 \in T$. We apply an IR engine twice to obtain two features. First, we use d_1 as a query and artifacts in T as the corpus. After running d_1 as a query through the IR engine, we capture the ranking of d_2 in the list of candidate links as the first feature. We then repeat the procedure. This time we consider d_2 as a query and S as the corpus, and capture the ranking of d_1 in the list of candidate links as the second feature. The reason for considering both directions is that previous work indicated that the choice of search direction affects search performance, especially traceability (Mills and Haiduc, 2017a).

3.2.2 Query quality features

While two IR ranking features provide information on textual similarity, previous studies have shown that the quality of query highly influences the results of IR (Mills et al., 2017). For example, if d_1 is of poor quality as a query, the IR ranking value may indicate that d_1 and d_2 are not linked, despite the fact that they are. To overcome this potential issue, we apply 16 QQ metrics to two documents in a possible link, which results in 32 different QQ features for each possible link. The QQ metrics can be divided into two categories: (1) specificity, i.e., how specific a query expression is; (2) coherence, which measures how focused a query is on a particular topic. The complete list of QQ features is given in the online appendix: http://github.com/TLR2019/resource/tree/master.

3.3 Data rebalancing

For each project, the number of valid traceability links is much smaller than that of invalid links. This imbalance of data can make it difficult to differentiate minority class samples (He and Garcia, 2009). To solve this problem, rebalancing techniques can be applied to the training set, which provides a more balanced representation of the majority and minority classes. Oversampling and undersampling are frequently used in sample rebalance methods. We use these two different methods in our experiments to determine if they could help improve the AL-based approach. Judging the experimental results, we choose the synthetic minority oversampling technique (SMOTE) as the rebalancing technique (Chawla et al., 2002). SMOTE is an oversampling method that adds samples by creating minority class samples. The process of creating new samples by SMOTE is as follows: for each sample x_i in the minority class, SMOTE searches for its nearest neighbors using the Euclidean distance and randomly selects one neighbor x_n. Then, a random number δ in [0, 1] is generated. The new sample x_{new} is created as

$$x_{\text{new}} = x_i + (x_n - x_i) \cdot \delta. \tag{1}$$

SMOTE can effectively solve sample imbalance by adding minority class samples.

4 Study design

For our research, we have three research goals. The first is to determine the best configuration for the AL-based approach. The second is to find a suitable training set size for the AL-based approach. The third is to compare the AL-based approach with the TSL-based approach. We primarily address the following questions:

1. What is the best configuration for the AL-based approach?

We aim to determine the combination of rebalancing techniques and classification algorithms to achieve the best performance of the AL-based approach. We measure performance in terms of F-score, calculated as the weighted average between precision and recall. The purpose of balancing precision and recall is to retrieve as many correct links as possible while keeping a low effort to discard false positives.

2. What should the size of the training set be for the AL-based approach?

To save manpower, the size of the training set should be as small as possible while ensuring the performance of the AL-based approach. We compare the AL-based approach with the IR-based approach (baseline) in terms of F-score.

3. Does the AL-based approach provide superior support for automatic traceability link recovery compared with the TSL-based approach?

Supposing the suitable training set size is N_{s}, we directly compare the AL- and TSL-based approaches in terms of F-score.

4.1 Experimental metrics

We use the traceability matrix (TM) to verify the correctness of traceability link classification. Table 1 shows four possible classification types.

Table 1 Traceability link classification

Classification	TM	Validation result	Correctness
Valid	Valid	TP	Correct
Invalid	Valid	FN	Incorrect
Valid	Invalid	FP	Incorrect
Invlaid	Invalid	TN	Correct

Among them, true positive (TP) and true negative (TN) represent the correct classification. False positive (FP) and false negative (FN) represent wrong classification. Precision and recall are two common metrics for measuring the performance of TLR approaches:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \times 100\%, \tag{2}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \times 100\%. \tag{3}$$

Precision is the ratio of the number of links correctly classified as valid to the number of links classified as valid, and recall is the ratio of the number of links correctly classified as valid to the number of valid links in TM. Precision and recall are orthogonal metrics used to measure two different concepts. An aggregate measure, F-score, is used to obtain a balance between them:

$$\text{F-score} = 2 \times \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{4}$$

We compare mainly the performance of different approaches in terms of F-score.

4.2 Data collection

To answer the three research questions mentioned earlier, we use seven datasets from five software projects. We apply the typical preprocessing steps employed in TLR on artifacts in these datasets (Kuang et al., 2017). The artifacts are normalized by standard preprocessing techniques, including splitting identifiers, special token elimination, stop word removal, and stemming. Table 2 depicts the dataset information, including the numbers of invalid and valid traceability links and artifact types in each dataset. Note that, as described in Section 3.3, the data is highly unbalanced and the average ratio of invalid to valid is approximately 11:1.

4.3 Baseline approach

We use an IR-based approach as the baseline approach. IR-based approaches require humans set a similarity threshold to select valid traceability links (Marcus and Maletic, 2003). Researchers have to determine an optimal threshold that retrieves as many correct links as possible while keeping a low effort to discard false positives. However, an optimal threshold is difficult to identify, as it may change with projects (Lucia et al., 2007). For each dataset, we choose a number from 0.1 to 0.9 that can achieve the highest F-score as the threshold. Traceability links with similarity above this threshold are deemed valid. There have already been some IR-based traceability tools that can be directly used, such as Retro (http://opensource.gsfc.nasa.gov/projects/RETRO) and Tracelab (http://www.coest.org/). IR used in our experiments is the vector space retrieval provided by Retro. Vector space retrieval treats each document in the document collection and each query as a set of keywords $(t_1, t_2, \ldots, t_n)$, where each keyword t_k $(k = 1, 2, \ldots, n)$ is given a weight w_k according to its importance in the document. Each document d_i is represented as a vector of keyword weights $\boldsymbol{d}_i = (w_{i1}, w_{i2}, \ldots, w_{in})$ and each query $\boldsymbol{q}$ is represented as $\boldsymbol{q} = (w_{q1}, w_{q2}, \ldots, w_{qn})$. The most commonly used weight estimation formula is

$$w_{ik} = \text{tf}_{ik} \cdot \text{idf}_k = \text{tf}_{ik} \cdot \left(\log\left(\frac{N}{n_k}\right) + 1\right), \tag{5}$$

where tf_{ik} is the frequency of the occurrence of keyword t_k in document d_i, N is the total number of documents in the collection, and n_k is the number of documents containing t_k. Given a document vector $\boldsymbol{d}_i$ and a query vector $\boldsymbol{q}$, the similarity between them is calculated as the cosine of the angle between the

Table 2 Datasets used in the evaluation

Dataset	Number of invalid links	Number of valid links	Artifact type	
			Source	Target
eAnci	7091	554	UC	CC
SMOS	5656	1044	UC	CC
MODIS	890	41	HighR	LowR
EasyClinic (TC-UC)	1827	63	TC	UC
EasyClinic (TC-CC)	2757	204	TC	CC
EasyClinic (ID-TC)	1177	83	ID	TC
eTour	6363	365	UC	CC
Total	25 761	2354	–	–

HighR: high-level requirements; LowR: low-level requirements; UC: use cases; CC: code classes; ID: interaction diagrams; TC: test cases two vectors:

$$\mathrm{sim}(\boldsymbol{d}_i, \boldsymbol{q}) = \cos(\boldsymbol{d}_i, \boldsymbol{q}) = \frac{\sum\limits_{k=1}^{n} w_{ik} w_{qk}}{\sqrt{\sum\limits_{k=1}^{n} w_{ik}^2 \sum\limits_{k=1}^{n} w_{qk}^2}}. \tag{6}$$

4.4 Answering question 1: determining the best configuration of the AL-based approach

We combine the commonly used rebalancing techniques with classification algorithms (Table 3) to implement the AL-based approach. We evaluate each implementation of the AL-based approach with every dataset independently, running 10-fold cross validations 10 times and averaging the results. The configuration that achieves the highest average F-score across all datasets is the best configuration. After determining the best configuration, we use the Mann-Whitney U test (a nonparametric test that does not assume samples from normally distributed populations) to calculate a statistically significant difference between the best configuration and other configurations.

Table 3 Classification algorithms and rebalancing techniques

Category	Variable	Note
Classification algorithms	RF	Classifier that uses a multitude of RF
	Naive Bayes	Naive Bayes classifier using estimator classes
	Logistic	Regression model with a ridge estimator
	SVM	Support vector machine classifier
Rebalancing techniques	SMOTE	Adding minority class samples
	Undersampling	Reducing majority class samples
	None	No rebalancing technique is applied

4.5 Answering question 2: determining a suitable training set size for the AL-based approach

We use training sets of different sizes to implement the AL-based approach. We set five training set sizes: 2%, 4%, 6%, 8%, and 10% of dataset size. We assume that the training set size is N. We randomly select $N/2$ samples for labeling to initialize a training set, and then select $N/2$ samples from the unlabeled sample set based on the pool-based sample selection strategy for labeling and add them to the training set. We aim to make the training set as small as possible and the trained classifier have high performance. We compare the AL-based approach using training sets of different sizes with the IR-based approach.

4.6 Answering question 3: comparing the AL-based approach with the TSL-based approach

We obtain a suitable training set size N_s in question 2. For the AL-based approach, we randomly select $N_\mathrm{s}/2$ samples for labeling to initialize a training set, and then label $N_\mathrm{s}/2$ samples selected by the pool-based sample selection strategy and add them to the training set. For the TSL-based approach, we randomly select N_s samples for labeling as a training set. The AL- and TSL-based approaches have training and test sets of the same size. We train a classifier on the training set and use the test set to test the performance of the classifier. Both the training sets of the AL- and TSL-based approaches are generated by random sampling. To mitigate the effect of sampling bias, we run 10 trials of each approach on each dataset and average the results. Finally, for each dataset, we compare the average F-score obtained by the two approaches and use the Mann-Whitney U test to calculate the significant difference (at the 0.05 significance level).

5 Results

5.1 Question 1: determining the best configuration of the AL-based approach

Table 4 shows the average F-score achieved by each combination of rebalancing techniques and classification algorithms across all of the datasets in this study. The configuration with the highest F-score is displayed in bold and chosen as a default configuration used in questions 2 and 3. The results show that RF significantly outperforms all other classification algorithms and achieves the best results when using SMOTE for data rebalancing. Therefore, the combination of RF and SMOTE is the best

Table 4 Average F-score achieved by the implementation of the AL-based approach across all datasets

Rebalancing technique	Average F-score (%) RF	Naive Bayes	Logistic	SVM
None	76.02	37.66*	46.00*	40.52*
SMOTE	**79.41**	35.02*	51.78*	49.25*
Undersampling	53.66*	32.46*	37.64*	36.63*

The bold font represents the best configuration. * performs statistically significantly worse than the best configuration (at the 0.05 significance level)

one. The RF classification algorithm with SMOTE rebalancing is the best configuration of the AL-based approach for all datasets in terms of F-score.

5.2 Question 2: determining a suitable training set size for the AL-based approach

Table 5 shows that for all datasets, as the size of the training set increases, the F-score of the AL-based approach also gradually increases. It is found that when we use 2% or 4% of the dataset as the training set, the performance of the AL-based approach does not significantly improve compared to that of the IR-based approach (the average F-score increases by no more than 6%). However, when we use 6%, 8%, or 10% of the dataset as the training set, the AL-based approach significantly outperforms the IR-based approach (the average F-score increases by more than 15%). In other words, using 6%, 8%, or 10% of the dataset as the training set could ensure the performance of the AL-based approach. We consider not only the performance of the AL-based approach, but also the size of the training set. To save manpower, the training set should be as small as possible. Therefore, 6% is more suitable than 8% or 10%. When we use 6% of the dataset as the training set, the AL-based approach outperforms the IR-based approach for all datasets except eTour. The AL-based approach improves the performance for six of the seven datasets by more than 11% in terms of F-score. For eTour, the F-score of the AL-based approach is only 6.6% lower than that of the IR-based approach. Overall, the AL-based approach using 6% of the dataset as the training set outperforms the IR-based approach by more than 15% in terms of the average F-score. The 6% dataset size is a suitable training set size for the AL-based approach.

5.3 Question 3: comparing AL- and TSL-based approaches

Table 6 shows a comparison of precision, recall, and F-score between AL- and TSL-based approaches for each dataset. When we use 6% of the dataset as the training set, the AL-based approach outperforms the TSL-based approach in terms of precision, recall, and F-score for each of the seven datasets. There are some differences in the performance improvements provided by the AL-based approach. For example, in the cases of SMOS and MODIS, the AL-based approach outperforms the TSL-based approach by less than 10% in terms of F-score. However, the AL-based approach improves the performance for EasyClinic (TC-UC) and EasyClinic (TC-CC) by more than 24% in terms of F-score. Overall, the AL-based approach outperforms the TSL-based approach by more than 16% in terms of the average F-score. Finally, the performance of the AL-based approach is statistically significantly better than that of the TSL-based approach at the 0.05 significance level in terms of F-score for each dataset. When we use 6% of the dataset as the training set, the AL-based approach significantly outperforms the TSL-based approach for all the seven datasets in terms of F-score.

6 Threats to validity

A possible threat is the randomness of experimental results. For the AL- and TSL-based approaches, we consider that good results might purely be obtained by chance through only one trial. Therefore, we run multiple trials of each approach on each dataset and then average the results. Finally, we use statistical tests to add rigor to the analysis of experimental results.

Another possible threat is the extent to which we can generalize the results of our studies. It is difficult to find a large software system with a large number of artifacts and a high-quality TM. Therefore, we cannot claim that our research results can be generalized to large software systems. However, our findings are still meaningful since we have evaluated seven datasets extracted from five different medium-sized software systems. Moreover, these datasets were created by domain experts and have been used many times in previous traceability studies.

Table 5 Average F-score achieved by the implementation of the AL-based approach using training sets of different sizes

Dataset	Average F-score (%)					
	AL					IR*
	2%	4%	6%	8%	10%	
eAnci (CC-UC)	24.24	50.07	55.05 (+30.50)	60.02	64.24	24.55
SMOS (CC-UC)	39.75	32.88	39.13 (+12.17)	40.47	42.62	26.96
MODIS (HighR-LowR)	26.77	29.13	38.18 (+11.31)	39.78	40.51	26.87
EasyClinic (TC-UC)	25.57	33.96	59.81 (+16.06)	62.89	75.65	43.75
EasyClinic (TC-CC)	51.97	66.01	76.56 (+29.82)	83.53	90.41	45.74
EasyClinic (ID-TC)	50.42	55.64	64.50 (+16.53)	74.39	81.45	47.97
eTour (CC-UC)	38.98	42.09	46.09 (−6.60)	47.48	47.83	52.69
Average	36.96	44.25	54.19 (+15.83)	58.37	63.24	38.36

* Baseline (IR: information retrieval). The number in the parentheses represents the difference of F-score between the AL- and IR-based approaches

Table 6 Precision, recall, and F-score of the AL- and TSL-based approaches

Dataset	Precision (%)		Recall (%)		F-score (%)	
	AL	TSL	AL	TSL	AL	TSL
eAnci (CC-UC)	73.37	50.95	44.05	37.27	55.05 (+12.00)	43.05
SMOS (CC-UC)	57.78	52.39	29.58	24.92	39.13 (+5.35)	33.78
MODIS (HighR-LowR)	72.41	32.04	25.93	30.69	38.18 (+6.83)	31.35
EasyClinic (TC-UC)	95.05	75.44	43.64	28.76	59.81 (+18.16)	41.65
EasyClinic (TC-CC)	89.97	71.45	66.63	40.29	76.56 (+25.03)	51.53
EasyClinic (ID-TC)	87.56	67.79	51.06	25.83	64.50 (+27.09)	37.41
eTour (CC-UC)	68.98	59.40	34.60	18.56	46.09 (+17.81)	28.28
Average	77.87	58.50	41.59	29.48	54.19 (+16.04)	38.15

The number in the parentheses represents the difference of F-score between the AL- and TSL-based approaches

7 Conclusions and future work

For projects without traceability information, the TSL-based approach for establishing traceability links consumes too much manpower. To solve this problem, we propose an AL-based approach for automatic TLR. First, we empirically derive the best configuration of the AL-based approach on seven datasets commonly used to evaluate new approaches to TLR. Then, we choose a suitable training set size for the AL-based approach. Finally, we compare the AL-based approach with the IR- and TSL-based approaches. It is shown that the AL-based approach outperforms the other two approaches in terms of F-score.

Because some IR-related features have been used, the AL-based approach also suffers from the "vocabulary mismatch" problem. Jin et al. (2017) proposed two query augmentation techniques to solve this problem: (1) replacing the original query terms with terms learned through web-mining; (2) using a domain ontology to augment query terms. In the future, we will improve the IR-related features using these techniques.

Contributors

Tian-bao DU designed the research. Guo-hua SHEN drafted the manuscript. Zhi-qiu HUANG, Yao-shen YU, and De-xiang WU helped organized the manuscript. Tian-bao DU revised and finalized the paper.

Compliance with ethics guidelines

Tian-bao DU, Guo-hua SHEN, Zhi-qiu HUANG, Yao-shen YU, and De-xiang WU declare that they have no conflict of interest.

References

Antoniol G, Canfora G, Lucia A, et al., 2000. Information retrieval models for recovering traceability links between code and documentation. 16th Int Conf on Software Maintenance, p.40-49.
https://doi.org/10.1109/ICSM.2000.883003

Asuncion HU, Asuncion AU, Taylor RN, 2010. Software traceability with topic modeling. 32nd Int Conf on Software Engineering, p.5-104.
https://doi.org/10.1145/1806799.1806817

Borg M, Runeson P, Ardö A, 2013. Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability. *Emp Softw Eng*, 19(6):565-1616.
https://doi.org/10.1007/s10664-013-9255-y

Breiman L, 2001. Random forests. *Mach Learn*, 45(1):5-32.
https://doi.org/10.1023/A:1010933404324

Chawla NV, Bowyer KW, Hall LO, et al., 2002. Smote: synthetic minority over-sampling technique. *J Artif Intell Res*, 16(1):321-357.
https://doi.org/10.1613/jair.953

Cheng Y, Chen ZZ, Liu L, et al., 2013. Feedback driven multiclass active learning for data streams. 22nd Int Conf on Information & Knowledge Management, p.1311-1320. https://doi.org/10.1145/2505515.2505528

Cleland-Huang J, Settimi R, Duan C, et al., 2005. Utilizing supporting evidence to improve dynamic requirements traceability. 13th Int Conf on Requirements Engineering, p.135-144. https://doi.org/10.1109/RE.2005.78

Cleland-Huang J, Settimi R, Zou XC, et al., 2007. Automated classification of non-functional requirements. *Req Eng*, 12(2):103-120.
https://doi.org/10.1007/s00766-007-0045-1

Cleland-Huang J, Czauderna A, Gibiec M, et al., 2010. A machine learning approach for tracing regulatory codes to product specific requirements. 32nd Int Conf on Software Engineering, p.155-164.
https://doi.org/10.1145/1806799.1806825

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

He H, Garcia E, 2009. Learning from imbalanced data. *IEEE Trans Knowl Data Eng*, 21(9):1263-1284.
https://doi.org/10.1109/TKDE.2008.239

Jin G, Gibiec M, Cleland-Huang J, 2017. Tackling the term-mismatch problem in automated trace retrieval. *Emp Softw Eng*, 22(3):1103-1142.
https://doi.org/10.1007/s10664-016-9479-8

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

Li ZH, Chen MR, Huang LG, et al., 2015. Recovering traceability links in requirements documents. 19th Conf on Computational Natural Language Learning, p.237-246. https://doi.org/10.18653/v1/K15-1024

Lucia A, Fasano F, Oliveto R, et al., 2007. Recovering traceability links in software artifact management systems using information retrieval methods. *ACM Trans Softw Eng Methodol*, 16(4):13.
https://doi.org/10.1145/1276933.1276934

Lucia A, Marcus A, Oliveto R, et al., 2012. Information retrieval methods for automated traceability recovery. In: Cleland-Huang J, Gotel O, Zisman A (Eds.), Software and Systems Traceability. Springer, London, p.71-98.
https://doi.org/10.1007/978-1-4471-2239-5

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

Marcus A, Maletic JI, Sergeyev A, 2005. Recovery of traceability links between software documentation and source code. *Int J Soft Eng Knowl Eng*, 15(5):811-836.
https://doi.org/10.1142/S0218194005002543

Mills C, Haiduc S, 2017a. The impact of retrieval direction on IR-based traceability link recovery. 39th Int Conf on Software Engineering: New Ideas and Emerging Technologies Results Track, p.51-54.
https://doi.org/10.1109/ICSE-NIER.2017.14

Mills C, Haiduc S, 2017b. A machine learning approach for determining the validity of traceability links. 39th Int Conf on Software Engineering Companion, p.121-123.
https://doi.org/10.1109/ICSE-C.2017.86

Mills C, Bavota G, Haiduc S, et al., 2017. Predicting query quality for applications of text retrieval to software engineering tasks. *ACM Trans Softw Eng Methodol*, 26(1):3.
https://doi.org/10.1145/3078841

Mills C, Escobar-Avila J, Haiduc S, 2018. Automatic traceability maintenance via machine learning classification. 34th Int Conf on Software Maintenance and Evolution, p.369-380. https://doi.org/10.1109/ICSME.2018.00045

Mirakhorli M, Shin Y, Cleland-Huang J, et al., 2012. A tactic-centric approach for automating traceability of quality concerns. 34th Int Conf on Software Engineering, p.639-649.
https://doi.org/10.1109/ICSE.2012.6227153

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

Rempel P, Mäder P, 2017. Preventing defects: the impact of requirements traceability completeness on software quality. *IEEE Trans Softw Eng*, 43(8):777-797.
https://doi.org/10.1109/TSE.2016.2622264