Adaptive User Feedback for IR-based Traceability Recovery

Annibale Panichella*, Andrea De Lucia†, Andy Zaidman*
* Delft University of Technology, The Netherlands
† Department of Management & Information Technology, University of Salerno, Italy
a.panichella@tudelft.nl, adelucia@unisa.it, a.e.zaidman@tudelft.nl

***Abstract*—Traceability recovery allows software engineers to understand the interconnections among software artefacts and, thus, it provides an important support to software maintenance activities. In the last decade, Information Retrieval (IR) has been widely adopted as core technology of semi-automatic tools to extract traceability links between artefacts according to their textual information. However, a widely known problem of IR-based methods is that some artefacts may share more words with non-related artefacts than with related ones.**

To overcome this problem, enhancing strategies have been proposed in literature. One of these strategies is *relevance feedback*, which allows to modify the textual similarity according to information about links classified by the users. Even though this technique is widely used for natural language documents, previous work has demonstrated that relevance feedback is not always useful for software artefacts.

In this paper, we propose an *adaptive* version of relevance feedback that, unlike the standard version, considers the characteristics of both (i) the software artefacts and (ii) the previously classified links for deciding whether and how to apply the feedback. An empirical evaluation conducted on three systems suggests that the adaptive relevance feedback outperforms both a pure IR-based method and the standard feedback.

***Index Terms*—Software Traceability, Information Retrieval, User Feedback Analysis, Empirical Software Engineering.**

I. Introduction

During the software development life cycle, developers produce several software artefacts other than source code, such as requirements, use cases, design documents, etc. These artefacts are bound to each other (e.g., because related to the same software modules) and, thus, during software maintenance activities developers have to manage their interconnections (links) [1]. Even if software engineers would benefit from traceability management in many tasks, such as *concept location* [2], *requirements tracing* [3], *impact analysis* [4] or *source code reuse* [5], [4], the manual identification of links is very expensive, error prone and infeasible especially in an industrial context [1], [6].

For these reasons, in recent years researchers have proposed the usage of Information Retrieval (IR) as medium to support the traceability recovery process [5], [6], [7]. Typically, an IR process extracts textual information from a software repository and compares a set of source artefacts (e.g., requirements) against another set of artefacts (e.g., source code). Pairs of artefacts having higher textual similarity are candidates to be true links because they share textual information. One of the main challenges of IR-based techniques consists in reducing the number of false positives, i.e., pairs of software artefacts having high textual similarity but that are not related to each other. As such, several enhancing strategies have been proposed in literature, such as the use of smoothing filters [8], external dictionary [9], identifier expansion techniques [10], or augmenting textual information with structural information [11].

Hayes *at al.* [3] proposed to use *relevance feedback* to improve the accuracy of vector based IR methods for requirement tracing by incorporating the judgments provided by the users for already classified links. For the Vector Space Model (VSM), this procedure is implemented using the Standard Rocchio algorithm [12] to increase the textual similarity between connected artefacts (correct links). Even if some studies [3], [13] demonstrated that relevance feedback can be helpful for requirement tracing, another work [14] demonstrated that relevance feedback does not improve and sometimes worsens the accuracy of an IR method when applied to different software artefacts.

We note that the standard Rocchio algorithm [12] has been used in previous work on traceability recovery in a similar manner to the way it is used for traditional IR tasks, with the assumption that software documents and natural language documents exhibit similar properties. For example, the Rocchio algorithm is generally used to augment the vocabulary of the queries, that in traditional IR problems contains only few words when compared with the size of the documents to retrieve [12]. In the context of traceability recovery this assumption is not true: source artefacts (queries) can contain more words than target artefacts (documents to retrieve). Moreover, recent research has also demonstrated that in general software artefacts are more homogeneous and predictable than natural language text [15], [16].

Stemming from the consideration that software documents presents specific peculiarities with respect to natural language documents, in this paper we introduce and propose an *adaptive* version of *relevance feedback* to overcome the limitations of the standard Rocchio algorithm [12] when applied to traceability recovery. We evaluated the proposed approach in an empirical study involving three software systems belonging to different domains (industrial and academic projects) and developed with different programming languages. The empirical evaluation is steered by the following research questions:

RQ$_1$ *Does the adaptive relevance feedback improve the* *performances of the Vector Space Model?*

RQ$_2$ *Does the adaptive relevance feedback outperform the standard relevance feedback?*

The results achieved reveal that, unlike the standard Rocchio algorithm, the *adaptive relevance feedback* statistically improves the performance of IR based traceability recovery.

The remainder of the paper is organized as follows. Section II provides background notions on IR-based traceability recovery and discusses related work. Section III introduces the proposed adaptive version of relevance feedback, while section IV describes the design of the empirical study we have performed to evaluate the benefits of the proposed adaptive approach. Results are reported and discussed in Section V, while Section VI discusses the threats to validity. Finally, Section VII concludes the paper.

II. Background and Related Work

This section summarises background notions about IR-based traceability recovery and discusses related work.

A. IR-based Traceability Recovery

Information Retrieval (IR) refers to a popular family of techniques for automatically extracting and managing textual information from documents, e.g., web pages, books, etc. Such techniques rely on the textual similarity among documents to derive their interconnections with the idea that documents sharing a large number of words (*terms*) have also high probability to share the same meaning. Even though different IR techniques exist, they follow a standard process that can be summarised in three steps: (i) *artefact-indexing*, (ii) building the *term-by-document matrix*, and (iii) computing the *textual similarity* using a specific formula.

During the first step, a textual *corpus* is built by extracting the keywords contained in the documents, removing common words (via *stop word function* and/or *stop word list*), *stemming* terms having the same root form and splitting composite identifiers [10], [17], [18]). Then, the collected data are stored in a $m \times n$ matrix, called the *term-by-document matrix* [19], where m is the number of terms that occur within artefacts, n is the number of artefacts, and the generic entry $w_{i,j}$ measures the relevance (i.e., weight) of the i^{th} term in the j^{th} document [19]. One of the most used weighting schemas is *term frequency - inverse document frequency* (*tf-idf*) [19], which attributes more importance to words having a high frequency in a document (high *tf*) and that are contained in few documents (high *idf*).

Finally, starting from the *term-by-document matrix*, a specific algebraic formula is used to compute the textual similarity of pairs of artifacts. The representation of the documents and the algebraic formula vary depending on the adopted IR method. The most used IR methods are (i) the probabilistic models [20], [21], (ii) the Vector Space Model (VSM) [19], and (iii) its extension called Latent Semantic Indexing (LSI) [22]. Further works have used different methods to recover traceability links between different types of artefacts [23], [24], [25], [26]. However, a recent empirical study showed that none of these techniques statistically outperforms the others [27]. For this reason, in this paper we consider only the Vector Space Model (VSM) to compare the textual similarity between different kinds of artefacts. In the VSM, artefacts are represented as vectors of terms occurring within artefacts in a repository, corresponding to column vectors of the term-by-document matrix [19], while the textual similarity among artefacts is measured as the cosine of the angle between the corresponding vectors.

A main challenge for IR-based traceability recovery methods is that a high textual similarity represents only a probability that pairs of source-target artefacts are linked. In order to improve the retrieval accuracy, researchers have proposed the use of enhancing strategies. Some previous work focused on artefact indexing [10], or proposed the usage of a new weighting schema [28], a project glossary [29], and an external dictionary [9]. Other works have suggested to use part-of-speech tagger to extract critical terms [15], or using a smoothing filter to automatically remove "common" words that do not help to characterise the artefacts content [8]. A detailed survey on IR-based techniques and enhancing strategies used in literature can be found in recent papers [6], [7].

B. Relevance feedback

A popular method to improve the performance of IR methods when dealing with natural language documents is represented by relevance feedback, i.e., incorporating the judgment provided by users to change the queries. In a traditional IR query context, the Standard Rocchio [12] is the classic algorithm to implement relevance feedback for VSM: it modifies a query vector on the basis of partial knowledge of known relevant and non-relevant documents provided by user. More formally, let $\overrightarrow{q}$ be the initial query vector, D_r be the set of known relevant documents and D_{nr} be the set of known non-relevant documents. Then, the new query $\overrightarrow{q_{new}}$ is computed as follows:

$$\overrightarrow{q_{new}} = \alpha \, \overrightarrow{q} + \beta \, \frac{1}{\mid D_r \mid} \sum_{d_j \in D_r} \overrightarrow{d_j} - \gamma \, \frac{1}{\mid D_{nr} \mid} \sum_{d_j \in D_{nr}} \overrightarrow{d_j} \qquad (1)$$

where α, β and γ are the weights to be assigned to the old query, relevant documents and non-relevant documents respectively. Intuitively, in this way the document vectors from the relevant documents are added to the initial query vector (*positive feedback*) while the vectors from the non-relevant document are subtracted (*negative feedback*). Previous works have also revealed that a reasonable configuration might be $\alpha = 1$, $\beta = 0.75$ and $\gamma = 0.25$ [14], [30], i.e. relevant documents are three times more important than irrelevant ones.

Hayes *at al.* [3] adopted the same strategy for IR-based requirement tracing. They proposed an interactive link recovery process with multiple steps: at each iteration, a tool returns the top links in the ranked list using a fixed cut point. The software engineer classifies these links either as actual links or as false positives, thus providing feedback to the tool. The source artefact is re-weighted, the ranked list is recomputed. The process is repeated for some iterations [3], [13] or until all links are retrieved [14]. In the preliminary study [3], relevance feedback were used to enhance VSM when retrieving links among high-level and low-level requirements and also using the standard Rocchio only for few iterations. The empirical results showed that relevance feedback can help in improving the performances of VSM. A further work [13] also reached similar conclusions but highlighted that relevance feedback can improve both in precision and recall only for the first few iterations. It is important to note that both [3] and [13] only investigated the usage of relevance feedback in the context of requirements tracing. However, when the goal is to retrieve all the links among different categories of software artefacts (such as use cases, UML diagrams, etc.), relevance feedback does not improve and sometimes worsens the performance of an IR method [14]. Moreover, their performance varies over different software datasets and over different recall thresholds for the same dataset [14].

In this paper we introduce and propose an adaptive version of the relevance feedback that considers the characteristics of both (i) the software artefacts and (ii) the labeled data (information about classified links) for deciding whether and how to apply the relevance feedback.

Algorithm 1 Adaptive Relevance Feedback (ARF)

```
1: $List \leftarrow$ initial ranked list of candidate links
2: *Classified* $\leftarrow \emptyset$ // set of classified links
3: **for** each artefact $i$ **do**
4:     $TP_i \leftarrow \emptyset$ // initialize the set of True Positive for $i$
5:     $FP_i \leftarrow \emptyset$ // initialize the set of False Positive for $i$
6: **end for**
7: **while** not (stopping criterion) **do**
8:     Get the link $(s,t)$ on top of $List$
9:     The user classifies $(s,t)$.
10:    *Classified* $\leftarrow Classified \bigcup \{(s,t)\}$
11:    **if** $(s,t)$ is correct **then**
12:        $\text{TP}_s \leftarrow \text{TP}_s \bigcup \{t\}$
13:        $\text{TP}_t \leftarrow \text{TP}_t \bigcup \{s\}$
14:    **else**
15:        $\text{FP}_s \leftarrow \text{FP}_s \bigcup \{t\}$
16:        $\text{FP}_t \leftarrow \text{FP}_t \bigcup \{s\}$
17:    **end if**
18:    Let $V_s$ be the set of terms in $s$
19:    Let $V_t$ be the set of terms in $t$
20:    **if** $|V_s| \leqslant |V_t|$ and $|TP_s| \geqslant |FP_s|$ **then**
21:        apply the standard Rocchio to $s$
22:    **else**
23:        **if** $|V_t| < |V_s|$ and $|TP_t| \geqslant |FP_t|$ **then**
24:            apply the standard Rocchio to $t$
25:        **end if**
26:    **end if**
27:    Recompute the ranked $List$ of links
28:    $List \leftarrow List-$ *Classified*
29: **end while**
```

III. Putting the User into the Loop: an Adaptive Approach

According to Manning *a*t al. [31], the success of relevance feedback depends on a number of assumptions. Firstly, the standard Rocchio is used for queries with few words when compared to the size of the documents to retrieve. Indeed, Equation 1 tends to increase the size of the queries by adding terms contained in the relevant documents since $\beta > \gamma$ [30], [31]. Secondly, relevance feedback requires relevant documents to be similar to each other [31], i.e., they should cluster in the vector space (*cluster hypothesis*). The approach does not work very well if the relevant documents (software artefacts) form a multimodal class, i.e., several clusters of documents within the vector space [31]. This happens when there is a mismatch between the query's vocabulary versus the documents' vocabulary [31].

In this paper we note that both assumptions reported above are not true in the context of traceability recovery. Indeed, some source artefacts (used as "queries") may contain more terms than target artefacts and, thus, the relevance feedback should not be applied unconditionally to the source artefacts as done in previous works [3], [13], [14]. Moreover, the cluster hypothesis does not hold, because, as reported in related literature (e.g., [17], [18]), source and target artefacts use different vocabularies, e.g., requirements use problem-domain terms, whereas the source code uses synonyms, abbreviations or technology-domain terms. Therefore, we conjecture that a proper application of relevance feedback to software artefacts has to consider these aspects.

Specifically, given a pair of source and target artefacts (s,t) classified by the user, we propose to apply the standard Rocchio to the artefact containing the lower number of unique terms, i.e., to the shortest (less verbose) artefact between s and t. Secondly, since the cluster hypothesis does not hold, clusters of related and non-related artefacts are not sharply distinct and, thus, the standard Rocchio is not able to properly move the query toward the related documents and far away from the non-related ones [19], [31]. In this scenario, the standard Rocchio has to rely on a set of labeled data (already classified links) containing more relevant artefacts (positive feedback) than non-relevant ones (negative feedback) [19], [31]. Indeed, while positive feedback can help to move the query in the direction of relevant artefacts, negative feedback moves the query far away from non-relevant documents but not necessarily closer to the more relevant ones. Hence, we propose to apply the relevance feedback if and only if the number of correct links (i.e., the number of related artefacts) is greater or equals to the number of false positives (i.e., the number of non related artefacts) already classified by the user for the shortest artefact between s and t (i.e., positive feedback is greater than negative feedback).

In summary, in this paper we propose an *adaptive relevance feedback* that considers the verbosity of the software artefacts and the number of already classified correct links and false positives for deciding whether and how to apply the relevance feedback. For completeness, Algorithm 1 reports the pseudo-code of the proposed *adaptive relevance feedback*. The algorithm starts with the generation of the list of candidate links ordered in descending order of textual similarity ($List$), obtained by applying all the steps reported in Section II-A and using VSM as IR method. In lines 2-6 the Algorithm initialises sets used to keep track of the classification made by the user. Specifically, *Classified* denotes the set of already classified links. TP_i represents the set of artefacts j such that link (i,j) or (j,i) has been classified by the user as true links, and FP_i is the set of artefacts j such that link (i,j) or (j,i)

TABLE I
CHARACTERISTICS OF THE SOFTWARE REPOSITORIES USED IN THE CASE STUDY.

System	Artifact			Langage
	Kind	N.	Total N.	
Easy-Clinic	Use Cases (UC)	30	160	Italian
	UML Interaction Diagram (ID)	20		
	Test Cases (TC)	63		
	Code Classes (CC)	47		
i-Trust	Use Cases (UC)	33	80	English
	JSP	47		
Modis	High Level Requirements (HLR)	19	68	English
	Low Level Requirements (LLR)	49		

has been classified as false positive by the user. Then, the loop between lines 7-29 describes the *adaptive relevance feedback*. At each iteration, the user classifies the first candidate link (s, t) on top of the ranked list and she judges it as correct link or as false positive (lines 8-9) in Algorithm 1. Then, t is stored as true link or false positive for s according to the judgement provided by the user (lines 11-17). Thus, the *if* condition in lines 20-21 verifies whether the source artefact s should be modified using the standard Rocchio. This happens if and only if s contains a lower number of unique terms than t, and if the amount of available positive feedback is greater than the amount of negative feedback, i.e., the number of true links for s is greater than its number of false positive ($|TP_s| \geqslant |FP_s|$). If the previous condition is not verified, the *if* condition in lines 23-25 checks whether the standard Rocchio can be applied to the target artefact t. In the end, the ranked $List$ is recomputed and the links already classified are removed from it (lines 27-28 in Algorithm 1). Finally, the loop is repeated until the user stops the retrieval process.

IV. EMPIRICAL STUDY DESIGN

This section describes the design of the empirical study that we conducted in order to evaluate the accuracy of the proposed *adaptive relevance feedback* for IR/based traceability recovery. The context of the study is represented by repositories of different software artefacts extracted from three software projects: Easy-Clinic, i-Trust and Modis. Easy-Clinic[1] is a software system developed by Master's students at the University of Salerno. i-Trust is a medical application used as a class project for Software Engineering courses at the North Carolina State University[2], while MODIS[1] is the open source Moderate Resolution Imaging Spectroradiometer (MODIS) developed by NASA. Table I reports the main characteristics as well as the number of different kinds of artifacts contained in these repositories. Each repository also contains the traceability matrix built and validated by the application developers. We consider such a matrix as the "oracle" to evaluate the accuracy of the different experimented traceability recovery methods.

We investigated the following two research questions:

RQ$_1$ *Does the adaptive relevance feedback improve the performances of the Vector Space Model?*

[1]http://www.coest.org/index.php/resources/dat-sets
[2]http://agile.csc.ncsu.edu/iTrust/wiki/doku.php?id=tracing

TABLE II
TRACING ACTIVITIES PERFORMED ON THE OBJECT REPOSITORIES.

Repository	Tracing Activities	# Correct Links	Description
	A_1	83	Tracing UC onto CC
Easy-Clinic	A_2	69	Tracing ID onto CC
	A_3	200	Tracing TC onto CC
i-Trust	A_4	58	Tracing UC onto JSP
Modis	A_5	26	Tracing HLR onto LLR

RQ$_2$ *Does the adaptive relevance feedback outperform the standard relevance feedback?*

Therefore, we analyse and compare the performances of a traceability recovery process based on Vector Space Model (VSM) when instantiated with the following variants: (i) *Adaptive feedback*: the relevance feedback scheme proposed in section III; (ii) *No feedback*: a traditional traceability recovery process is instantiated without using any relevance feedback; (iii) *Standard feedback*: it is the traditional relevance feedback scheme based on the Standard Rocchio and used in previous work [3], [14].

We have used the *tf-idf* weight model, porter stemmer and a common stop word list, which in addition to standard Italian and English stop words included (i) programming language (C/Java) keywords, (ii) recurring words in document templates (e.g., use case, requirement, or test case template). To enrich the generalisability of our results we carry out five different traceability recovery activities, three on Easy-Clinic and one on the other repositories (see Table II). To evaluate the different traceability recovery methods, we first compare the accuracy of the different treatments at each point of the ranked list using two widely used IR metrics, namely *recall* and *precision*:

$$recall = \frac{|corr \cap retr|}{|corr|}\% \quad precision = \frac{|corr \cap retr|}{|retr|}\%$$

where *corr* and *retr* represent the set of correct links and the set of links retrieved by the tool respectively.

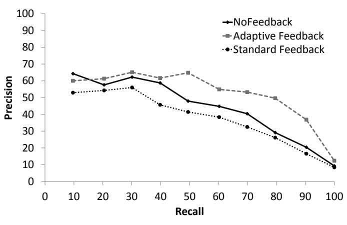
To provide statistical support for our findings we also use statistical tests to verify whether the number of false positives retrieved by VSM with adaptive relevance feedback is significantly lower than the number of false positives retrieved when using the standard Rocchio or without using any relevance feedback scheme. Since the number of correct links is the same when comparing different methods on the same traceability recovery activity (i.e., the data is paired), we use the one-tailed Wilcoxon Rank Sum test [32] with a p-value threshold of $\alpha = 5\%$. We also estimate the magnitude of the improvement in terms of false positives reduction when using the adaptive approach using the Cliff's Delta (d) [33], a non-parametric effect size measure for ordinal data whose score ranges in the interval $[-1; 1]$. It is equal to $+1$ when all values of one group are higher than the values of the other group and -1 when the reverse is true. Two overlapping distributions would have a Cliff's Delta equal to zero. The effect size is considered small for $0.148 \leq |d| < 0.33$, medium for $0.33 \leq |d| < 0.474$ and large for $|d| \geq 0.474$ [33].

TABLE III
IMPROVEMENT OF PRECISION AND FALSE POSITIVES REDUCTION ACHIEVED BY THE ADAPTIVE FEEDBACK AT DIFFERENT LEVELS OF RECALL.

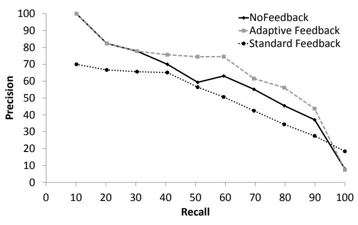
System	Traced artifacts	Rec 20%		Rec 40%		Rec 60%		Rec 80%		Rec 100%	
		Prec	FP	Prec	FP	Prec	FP	Prec	FP	Prec	FP
EasyClinic	UC→CC	**+3.71%**	**-14%**	**+2.94%**	**-11%**	**+10.10%**	**-33%**	**+20.53%**	**-58%**	**+3.08%**	**-27%**
	ID→CC	-	-	**+5.68%**	**-25%**	**+11.47%**	**-42%**	**+10.67%**	**-35%**	-0.51%	+7%
	TC→CC	**+39.13%**	**-88%**	**+47.27%**	**-90%**	**+58.84%**	**-94%**	**+59.03%**	**-94%**	**+6.23%**	**-30%**
i-Trust	UC→JSP	-	-	**+13.27%**	**-41%**	**+9.07%**	**-35%**	-0.07%	+1%	**+0.18%**	**-3%**
Modis	HLR→LLR	-	-	**+14.20%**	**-45%**	**+29.71%**	**-77%**	**+10.34%**	**-47%**	**+2.06%**	**-25%**

TABLE IV
IMPROVEMENT OF PRECISION AND FALSE POSITIVES REDUCTION ACHIEVED BY THE STANDARD ROCCHIO AT DIFFERENT LEVELS OF RECALL.

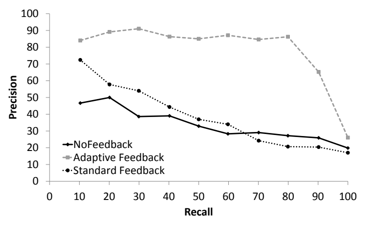
System	Traced artifacts	Rec 20%		Rec 40%		Rec 60%		Rec 80%		Rec 100%	
		Prec	FP	Prec	FP	Prec	FP	Prec	FP	Prec	FP
EasyClinic	UC→CC	-3.29%	+14%	-13.05%	+69%	-6.44%	+30%	-2.99%	+16%	-0.87%	+11%
	ID→CC	-15.69%	+133%	-4.88%	+25%	-12.46%	+67%	-11.08%	+59%	**+10.43%**	**-61%**
	TC→CC	**+7.75%**	**-27%**	**+5.28%**	**-20%**	**+5.68%**	**-23%**	-6.53%	+43%	-2.81%	+21%
i-Trust	UC→JSP	-40.00%	+100%	**+13.27%**	**-41%**	**+18.38%**	**-56%**	-3.38%	+41%	-2.74%	+72%
Modis	HLR→LLR	-16.67%	+100%	**+14.20%**	**-45%**	**+11.59%**	**-50%**	**+9.09%**	**-43%**	**+1.61%**	**-21%**



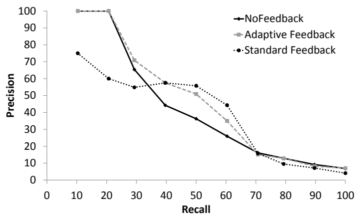
(a) Easy-Clinic: tracing UC onto CC



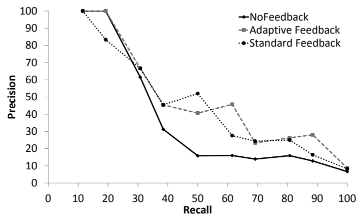
(b) Easy-Clinic: tracing ID onto CC



(c) Easy-Clinic: tracing TC onto CC



(d) i-Trust: tracing UC onto JSP



(e) Modis: tracing HLR onto LLR

Fig. 1. Precision-Recall curves for the different datasets.

V. EMPIRICAL STUDY RESULTS

Table III reports the improvements of precision and the reduction of retrieved false positives achieved using the proposed adaptive feedback over VSM at different levels of recall. A preliminary analysis shows that the adaptive feedback always achieved better recovery accuracy than that obtained without using any relevance feedback schema. In several cases for a recall level smaller than 80%, the adaptive feedback allows to achieve an improvement of precision ranging between 10% and 59% and a considerable reduction of retrieved false positives ranging between 8% and 94%. From the perspective of a software engineer that has to inspect the ranked list of candidate traceability links, such a result represents a substantial improvement in terms of false positive reduction. For example, using the adaptive feedback to trace test cases onto code classes in Easy-Clinic, it is possible to trace 102 correct links (about 50% of recall) discarding only 18 false positives (i.e. about 85% of precision). With the standard VSM model (without feedback) the software engineer should discard 208 false positive (i.e. 190 further false positives) for achieving the same level of recall. This result is impressive when comparing with the empirical results achieved by previous work [8] where improving the recovery accuracy when tracing test cases onto code classes turned out to be very difficult with other enhancing strategies. A less evident improvement is achieved for a recall percentile of 100%. In this case, the improvement is lower than 7% in terms of precision and lower than 30% in terms of false positives. The only exception to the rule is represented by Easy-Clinic when tracing UML diagrams onto code classes. This result confirms that, when the goal is to recover all correct links, there is an upper bound for performance improvements that is very difficult to overcome [14].

Table IV reports the differences in terms of precision and retrieved false positives achieved by VSM when using iteratively the standard Rocchio [14] and without the adaptive

TABLE V
RESULTS OF THE WILCOXON TESTS.

Comparison	A_1	A_2	A_3	A_4	A_5
Stand. Feedback Vs. No Feedback	1	1	0.95	0.95	**<0.01**
Adapt. Feedback Vs. No Feedback	**<0.01**	**<0.01**	**<0.01**	0.22	**<0.01**
Adapt. Feedback Vs. St. Feedback	**<0.01**	**<0.01**	**<0.01**	**<0.01**	0.32

TABLE VI
RESULTS OF CLIFF'S d EFFECT SIZE.

Comparison	A_1	A_2	A_3	A_4	A_5
Stand. Feedback Vs. No Feedback	-0.48	-0.13	-0.25	-0.21	**0.75** (L)
Adapt. Feedback Vs. No Feedback	**0.50** (L)	**0.48** (L)	**0.80** (L)	**0.02**	**0.70** (L)
Adapt. Feedback Vs. St. Feedback	**0.27** (L)	**0.24** (S)	**0.65** (L)	**0.3** (S)	**0.29** (S)

scheme proposed in this paper. The results show that the standard feedback improves the performance of VSM for MODIS (i.e., when tracing high-level requirements onto low-level requirements) when the recall is greater than 20%. This finding confirms the empirical results achieved by previous works [3], [13] where the application of standard feedback turned out to be effective for requirement tracing. However, when the goal is to retrieve traceability links among other kinds of artefact the application of the standard feedback sometimes improves and sometimes worsens the performance of VSM at different levels of recall as previously reported in [14]. Indeed, for the other datasets we can observe how the Rocchio algorithm achieves a variation of precision ranging between -40% and +18% and a variation of retrieved false positive ranging between -27% and +133%.

From the comparison of the results reported in Tables III and IV we notice that, unlike the standard feedback, the proposed *adaptive relevance feedback* allows to improve the performances of VSM not only for requirements but also for other software artefacts. This is confirmed by the graphical comparison reported in Figure 1, which plots the precision-recall graphs for VSM (*no feedback*), VSM with adaptive relevance feedback (*adaptive feedback*) and VSM with standard Rocchio (*standard feedback*). The improvement gained by the *adaptive feedback* is particularly evident for EasyClinic, while for MODIS and i-Trust there is more interleaving between the two feedback strategies. In particular, for i-Trust the adaptive feedback is substantially better than the standard feedback only for recall lower than 50%, while for MODIS the two corresponding precision-recall curves are very similar. To provide a deeper analysis, Table V shows the p-values of the Wilcoxon test (using Holm's correction) while Table VI illustrates Cliff's d values for all pairwise comparisons. Values for which the Wilcoxon test indicated a significant difference are shown in bold face. It can be noted that in 4 out of 5 cases the adaptive feedback provides a significant false positives reduction (p-values < 0.05) with a *large* effect size. The only exception to the rule is represented by the traceability activity A_4, i.e. when tracing use cases onto *jsp* files for i-Trust (in this case there is no statistical difference between the two treatments *adaptive feedback* and *no feedback*). In only 1 out of 5 cases, the standard feedback (i.e., Rocchio algorithm) is able to statistically improve the recovery accuracy of a standard traceability activity (i.e. *no feedback*). Finally, the adaptive feedback statistically outperforms (in terms of false positives reduction) the standard feedback in 4 out of 5 cases. Such a scenario is confirmed by analysing the effect size: it is *large* in 2 out of 5 cases and *small* in the remaining 3 cases.

VI. THREATS TO VALIDITY

This section discusses the threats that can affect the validity of our empirical study. For what concerns the *construct validity*, we used two widely adopted metrics, i.e. precision and recall, as well as the number of false positives for measuring the performance of VSM and its improvement when applying two different feedback strategies. Another important threat regards the accuracy of the oracle (traceability matrix) used to evaluate the results of all traceability activities. We used original traceability matrices provided by the software developers to mitigate such a threat.

For what concerns *internal validity*, there are several factors that can affect our results: (i) the language used to write the software artefacts, (ii) the kind of the artefacts, and (iii) the pre-processing used during the artefacts indexing. We have mitigated the former threat by selecting three software projects, one written in Italian and two written in English, that contain different kinds of software artefacts. The empirical results revealed that the adaptive relevance feedback improves the accuracy of VSM independently of the artefact's language and the kind of the artefacts to be traced. About the latter threat, we have mitigated it by showing that the adaptive feedback obtains a significant improvement when the corpus is pruned by stop word list, stemmer and the *tf-idf* weighting schema.

About the *conclusion validity* we support our empirical findings by using a proper statistical test, i.e the Wilcoxon non-parametric test, and a non-parametric effect size measure (Cliff's Delta) to provide a practical measurement of the magnitude of improvements. Beforehand, we verified the non-normality nature of the distributions using Shapiro's normality test. Since multiple tests were performed on the same data sets, we use the Holm's correction to correct p-values.

Potential threats to *external validity* can be related to the generalisation of our findings. With the aim of making our results as generalisable as possible, we have considered two academic (EasyClinic, i-Trust), and one industrial (Modis) projects. In addition, all repositories have previously been used by other authors to evaluate IR methods and other enhancing strategies [3], [8], [9], [15], [23], [25], [34].

VII. CONCLUSION

In this paper we propose *adaptive relevance feedback* to improve the performances of IR-based traceability recovery methods [5]. Relevance feedback has been previously proposed in literature [3], [13], [14], but these studies have reported contrasting results, showing that relevance feedback does not always improve the accuracy of IR methods, such as VSM, when applied to software artefacts [14]. In this paper we introduce an *adaptive* version of relevance feedback built upon the consideration that software artefacts do not share the same properties of natural language documents, on which the standard feedback relies. The results of the empirical study we conducted on three software systems libraries can be summarised as follows:

RQ$_1$ *Does the adaptive relevance feedback improve the performances of the Vector Space Model?* The proposed *adaptive relevance feedback* allows to gain an improvement of precision ranging between 10% and 59% with a considerable reduction of retrieved false positive raging between 8% and 95%. These improvements are completely independent from the artefact's language and the kind of software artefacts to be traced.

RQ$_2$ *Does the adaptive relevance feedback outperform the standard relevance feedback?* The *adaptive feedback* yielded strong, statistically significant improvements with respect to the standard feedback, which provides benefits only for requirement tracing. Specifically, the precision was significantly higher in 100% of cases where the goal was to retrieve links among high level software artefacts and source code files.

Future work will be devoted to corroborating the empirical findings reported in this paper by replicating the study over a larger number of software projects. We also plan to investigate the benefits that the proposed adaptive feedback might provide for other IR-methods, such as Latent Semantic Indexing (LSI), or in combination with other enhancing strategies, such as combining textual and structural information [11].

References

[1] J. Cleland-Huang and C. K. Chang, "Event-based traceability for managing evolutionary change," *IEEE Trans. on Software Engineering*, vol. 29, no. 9, pp. 796–810, 2003.

[2] D. Poshyvanyk and D. Marcus, "Combining formal concept analysis with information retrieval for concept location in source code," in *Proc. Int'l Conf. on Program Comprehension (ICPC)*. IEEE, 2007, pp. 37–48.

[3] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, "Advancing candidate link generation for requirements tracing: The study of methods." *IEEE Trans. on Software Engineering*, vol. 32, no. 1, pp. 4–19, 2006.

[4] A. De Lucia, F. Fasano, and R. Oliveto, "Traceability management for impact analysis," in *Proceedings of Frontiers of Software Maintenance*. IEEE, 2008, pp. 21–30.

[5] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, "Recovering traceability links between code and documentation," *IEEE Trans. on Software Engineering*, vol. 28, no. 10, pp. 970–983, 2002.

[6] M. Borg, P. Runeson, and A. Ardö, "Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability," *Empirical Software Engineering*, vol. 19, no. 6, pp. 1565–1616, 2014.

[7] A. De Lucia, A. Marcus, R. Oliveto, and D. Poshyvanyk, "Information retrieval methods for automated traceability recovery," in *Software and Systems Traceability*, J. Cleland-Huang, O. Gotel, and A. Zisman, Eds. Springer London, 2012, pp. 71–98.

[8] A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella, "Applying a smoothing filter to improve ir-based traceability recovery processes: An empirical investigation," *Information & Software Technology*, vol. 55, no. 4, pp. 741–754, 2013.

[9] J. H. Hayes, A. Dekhtyar, and J. Osborne, "Improving requirements tracing via information retrieval," in *Proc. International Requirements Engineering Conference*. IEEE, 2003, pp. 138–147.

[10] L. Guerrouj, M. Di Penta, G. Antoniol, and Y.-G. Guéhéneuc, "TIDIER: an identifier splitting approach using speech recognition techniques," *Journal of Software: Evolution and Process*, vol. 25, no. 6, pp. 579–599, 2011.

[11] A. Panichella, C. McMillan, E. Moritz, D. Palmieri, R. Oliveto, D. Poshyvanyk, and A. De Lucia, "When and how using structural information to improve ir-based traceability recovery," in *Proc. Conf. Software Maintenance and Reengineering (CSMR)*, 2013, pp. 199—208.

[12] J. Rocchio, *Relevance feedback in information retrieval*, G. Salton, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1971.

[13] L. Kong, J. Li, Y. Li, Y. Yang, and Q. Wang, "A requirement traceability refinement method based on relevance feedback." in *Proceedings of the 21st International Conference on Software Engineering & Knowledge Engineering (SEKE)*, 2009, pp. 37–42.

[14] A. De Lucia, R. Oliveto, and P. Sgueglia, "Incremental approach and user feedbacks: a silver bullet for traceability recovery," in *Proc. Int'l Conf. on Software Maintenance (ICSM)*. IEEE, 2006, pp. 299–309.

[15] G. Capobianco, A. De Lucia, R. Oliveto, A. Panichella, and S. Panichella, "On the role of the nouns in IR-based traceability recovery," in *Proc. Int'l Conf. on Program Comprehension (ICPC)*. IEEE, 2009, pp. 148–157.

[16] A. Hindle, E. T. Barr, Z. Su, M. Gabel, and P. T. Devanbu, "On the naturalness of software," in *Proc. Int'l Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 837–847.

[17] D. J. Lawrie, D. Binkley, and C. Morrell, "Normalizing source code vocabulary," in *Proc. Working Conference on Reverse Engineering (WCRE)*. IEEE, 2010, pp. 3–12.

[18] D. Lawrie and D. Binkley, "Expanding identifiers to normalize source code vocabulary," in *Proc. International Conference on Software Maintenance (ICSM)*. IEEE, 2011, pp. 113–122.

[19] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison-Wesley, 1999.

[20] G. Antoniol, G. Canfora, A. De Lucia, and E. Merlo, "Recovering code to documentation links in OO systems," in *Proc. Working Conference on Reverse Engineering (WCRE)*. IEEE CS Press, 1999, pp. 136–144.

[21] J. Cleland-Huang, R. Settimi, C. Duan, and X. Zou, "Utilizing supporting evidence to improve dynamic requirements traceability," in *Proc. Int'l Requirements Engineering Conference*. IEEE, 2005, pp. 135–144.

[22] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[23] H. U. Asuncion, A. Asuncion, and R. N. Taylor, "Software traceability with topic modeling," in *Proc. Int'l Conference on Software Engineering (ICSE)*. ACM, 2010, pp. 95–104.

[24] A. Abadi, M. Nisenson, and Y. Simionovici, "A traceability technique for specifications," in *Proc. Int'l Conference on Program Comprehension (ICPC)*. IEEE, 2008, pp. 103–112.

[25] G. Capobianco, A. De Lucia, R. Oliveto, A. Panichella, and S. Panichella, "Traceability recovery using numerical analysis," in *Working Conf. Reverse Engineering (WCRE)*. IEEE, 2009, pp. 195–204.

[26] P. Heck and A. Zaidman, "Horizontal traceability for just-in-time requirements: the case for open source feature requests," *Journal of Software: Evolution and Process*, vol. 26, no. 12, pp. 1280–1296, 2014.

[27] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia, "On the equivalence of information retrieval methods for automated traceability link recovery," in *Proc. Int'l Conf Program Comprehension (ICPC)*, 2010, pp. 68–71.

[28] R. Settimi, J. Cleland-Huang, O. Ben Khadra, J. Mody, W. Lukasik, and C. De Palma, "Supporting software evolution through dynamically retrieving traces to UML artifacts," in *Proc. Int'l Workshop on Principles of Software Evolution (IWPSE)*. IEEE, 2004, pp. 49–54.

[29] X. Zou, R. Settimi, and J. Cleland-Huang, "Improving automated requirements trace retrieval: a study of term-based enhancement methods," *Empirical Software Engineering*, vol. 15, no. 2, pp. 119–146, 2010.

[30] G. Salton and C. Buckley, "Improving retrieval performance by relevance feedback," *Journal of the American Society for Information Science*, vol. 41, pp. 288–297, 1990.

[31] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[32] W. J. Conover, *Practical Nonparametric Statistics*, 3rd ed. Wiley, 1998.

[33] R. J. Grissom and J. J. Kim, *Effect sizes for research: A broad practical approach*, 2nd ed. Lawrence Earlbaum Associates, 2005.

[34] J. Cleland-Huang, O. Gotel, and A. Zisman (eds.), *Software and Systems Traceability*. Springer-Verlag, 2011.