Technique Integration for Requirements Assessment

Alex Dekhtyar+ Jane Huffman Hayes+ Senthil Sundaram+
Ashlee Holbrook+ Olga Dekhtyar*
+*Department of Computer Science* **Institute For HIV Prevention*
{dekhtyar,hayes}@cs.uky.edu, {skart2, ashlee}@uky.edu, odekh2@uky.edu
University of Kentucky

Abstract

In determining whether to permit a safety-critical software system to be certified and in performing independent verification and validation (IV&V) of safety- or mission-critical systems, the requirements traceability matrix (RTM) delivered by the developer must be assessed for accuracy. The current state of the practice is to perform this work manually, or with the help of general-purpose tools such as word processors and spreadsheets Such work is error-prone and person-power intensive. In this paper, we extend our prior work in application of Information Retrieval (IR) methods for candidate link generation to the problem of RTM accuracy assessment. We build voting committees from five IR methods, and use a variety of voting schemes to accept or reject links from given candidate RTMs. We report on the results of two experiments. In the first experiment, we used 25 candidate RTMs built by human analysts for a small tracing task involving a portion of a NASA scientific instrument specification. In the second experiment, we randomly seeded faults in the RTM for the entire specification. Results of the experiments are presented.

1. Introduction

Developers who build software systems and "sell off" their software to the buyer must prove that their systems satisfy contractual requirements. When obtaining system certification (safety, security, etc.), the developers again must demonstrate or validate that the software meets the certification requirements. Reverse engineers who seek to extract requirements from a set of design elements and code must validate that the requirements have been satisfied by the as-built system. A similar task has to be performed by an IV&V analyst who must verify that the artifacts of the current phase of the development lifecycle properly satisfy the artifacts of the previous phase (e.g., that the design satisfies the requirements). Software maintainers deciding whether to reuse a component or to write it from scratch need to know if the component correctly satisfies its requirements.

The common thread (the unifying theme) of all the above situations is the need to know the mapping between the various artifact levels and the need to have a way to determine if this mapping is correct – a way to assess the mapping. The various professionals (IV&V analyst, reverse engineer, etc.) may have different names for it, but basically they all are determining the accuracy of the mapping. We illustrate this with a scenario where a requirements traceability matrix, or RTM, is used to map the elements of a pair of artifacts to each other (for example, a design specification's elements are mapped to code components). Let us call the *i*th element of the design document $\boldsymbol{d_i}$ and the *j*th component of the code $\boldsymbol{c_j}$. An RTM from the design to the code will be a collection of lists or entries for each item $\boldsymbol{d_i}$ that may contain from 0 to *N* $\boldsymbol{c_j}$ elements. In such an RTM, for each $\boldsymbol{d_i}$ entry, its candidate links list should be evaluated for *accuracy* – do we agree that all listed $\boldsymbol{c_j}$ elements are related to $\boldsymbol{d_i}$? It should also be checked for *completeness* – do we agree that all the $\boldsymbol{c_j}$ elements related to $\boldsymbol{d_i}$ are listed? In this work, we concentrate on accuracy.

The current ***state of the practice*** is to perform such evaluation manually, or with the help of general-purpose tools such as word processors, spreadsheets, and database management systems. Thus, the task is often not performed, or a "spot check" is performed rather than a more complete evaluation. An automated technique for this would allow the work to be performed thoroughly, and with much less human effort and error. The current ***state of research*** in the area of traceability [9,10,1,12,8] concentrates on trace recovery, rather than on RTM assessment tasks.

In this paper, we extend our previous work on trace recovery to the complementary problem of *RTM assessment.* To our knowledge, this paper represents the first attempt to address this problem in an automated fashion. We adapt prior trace recovery methods [9,10] and some newly introduced trace

recovery methods to the problem of determining whether the candidate links of a given (candidate) RTM should be kept in the RTM or rejected. We use *committees* of trace recovery methods, and let the committees *vote* on each individual candidate link.

This paper describes the results obtained from two studies. In one study we used a small dataset and candidate RTMs generated by human analysts. Our interest was to see if assessment by committee could detect mistakes that the humans had introduced into the RTMs. In the other study we looked at a larger dataset, and we *seeded* false positive links in the candidate RTMs using two seeding procedures. The question for this study is whether assessment by committee can detect mistakes made by one automated trace recovery method. Our *committees* work as simple unsupervised classifiers. We evaluate their work by measuring the *false positive detection rate* and the *correct link rejection rate*, the standard measures for assessing the quality of classifiers.

The paper is organized as follows. Section 2 presents related work. Section 3 presents our methodology for combining techniques for assessment of requirements mappings or RTMs. Section 4 describes our validation. Section 5 presents the results, and in Section 6 we present a discussion and future work.

2. Related work

There are two aspects to the study of automating requirements traceability: study of methods for automating traceability link recovery and other traceability tasks, and study of the impact of such methods on the entire tracing process and its efficiency. In [17] we have discussed the need for research on the latter problem: the study of the impact must involve studying how analysts use automated traceability tools/methods *in vivo*. However, before such studies can take place, good tools/methods must be discovered. This paper falls in the realm of the study of methods: looking for the means of giving analysts accurate assessment results. Individual information retrieval methods have shown promise for generating mappings – term frequency-inverse document frequency and probabilistic retrieval have been used by Antoniol et al. [1], latent semantic indexing has been used by Marcus and Maletic [12] and Hayes et al. [10]. In general, these methods, working alone, result in finding many of the related items (high recall), but also tend to retrieve many elements that are not relevant (low precision). We are not aware of any work on the assessment of requirements mappings.

The idea behind N-version programming [5] is that if a piece of code is required to perform reliably and correctly (based on some specification), the software is implemented in several programming languages and/or using orthogonal programming paradigms (such as structured and object-oriented). Theoretically, these diverse implementations should not fail in the same way. Then, each implementation "weighs in" or "votes" by executing and providing output. If a majority of the implementations generate the same output (agree), the output is returned and the result is deemed "correct." If the implementations cannot come to a majority ruling, the results are not "trusted."

Poshyvanyk et al also used the approach of combined experts to perform feature identification for maintenance programming support [15]. They found that combining scenario based probabilistic ranking and latent semantic indexing improved feature location when compared to the individual techniques. Based on the above related work, it appears that combining techniques might have promise for evaluating the accuracy of RTMs. Information retrieval methods have been shown to be helpful in generating or building such mappings, but can they assist in evaluating mappings and is it useful to use more than one technique for the assessment?

3. Methodology

The *RTM Assessment* problem studied can be expressed as follows:

> *Given two textual artifacts (referred to as a high-level and a low-level artifact), broken into individual elements, and a candidate Requirements Tracing Matrix (RTM) M, determine, for each candidate link in M, if it has been correctly included.*

To convert any trace recovery method into an RTM assessment method it is run on the given pair of artifacts, and the resulting candidate RTM *recovered by the method* is compared to the input RTM. The decision rule is: *a link from the input RTM is kept iff it is found in the RTM constructed by the method.* We extend this idea to the situation when more than one trace recovery method is available.

For the purpose of this paper, we define a *trace recovery method* as any algorithm that, given a pair of textual artifacts broken into elements, outputs a candidate RTM for them. Many of the trace recovery methods discussed below provide some extra

information about the recovered links – e.g., the perceived *similarity* between the two linked elements. However, for constructing RTM assessment methods discussed here, we view each trace recovery method as a *filter*: for each pair of high- and low-level elements, the trace recovery method decides whether it is included in the output, or whether it is rejected. Each trace recovery method *F* can be viewed as a function: $F: H \times L \rightarrow \{True, False\}$, where $F(i,j)=True$ means that the method believes that the high-level element h_i traces to low-level element l_j.

Suppose, we are given a collection *F1,…,Fk* of trace recovery methods. To construct an RTM assessment algorithm that uses *F1,…,Fk,* , we first need to select a *voting scheme*, i.e., a decision rule which will determine if a link is reported or rejected. We consider only simple decision rules that treat each method *F1,…,Fk,* equally, and simply count, for each link *(i,j),* the total number of *Fl(i,j)* which evaluate to *True*.

Given an RTM M={(i,j)}, the RTM assessment method based on F1,..,Fk using the decision rule r works as follows: each candidate link (i,j) from M is, in turn, evaluated by F1,..,Fk. If the number of methods evaluating to True exceeds the threshold of r, the link (i,j) is kept in the output RTM. Otherwise, (i,j) is rejected. For example, a *majority rule* accepts a link if it is recovered by more than one half of all available methods.

In this paper, we consider an RTM assessment committee composed of three, four and five different trace recovery methods, and employ a number of different decision rules. We briefly describe the five trace recovery methods next.

Vector Space Retrieval using tf-idf keyword weighting (td-idf). This trace recovery method, described in detail in [9,10], represents each element as a vector of keyword weights. It uses *term frequency* in the document and *inverse document frequency,* which measures the rarity of a keyword/term in the entire artifact, to compute weights of individual keywords. Similarity between two vectors is computed as a cosine of the angle between them (i.e., as the normalized dot-product of the vectors).

χ^2-based Keyword Extraction (KE). This method, introduced recently to trace recovery [11], is an extension of a single-document keyword extraction method of Matsuo and Ishizuka [13], which turns it into a retrieval method. All keywords in the entire artifact are ranked based on their perceived importance[1], and only the top X% of the keywords are used in the actual retrieval. We used the vectors of keyword weights generated by the tf-idf method, set X=50%, and projected out all keywords in the bottom half of the produced ranked list.

Probabilistic Information Retrieval (ProbIR). We used the *binary independence retrieval* method, which tries to estimate the odds ratio of a pair of elements being a link vs. not being a link, and is often referred to as "probabilistic retrieval" [3]. The quality of the results returned by this method on small datasets[2] depends highly on the initial estimates of two quantities: probability of a true link containing a specific keyword, and probability of a false positive link containing a specific keyword[3].

Latent Semantic Indexing (LSI). LSI transforms the element-by-keyword representation of an artifact into an element-by-*latent topic* representation using singular-valued matrix decomposition [6]. The key difference between the original and final representation is the fact that the number of *latent topics* can be significantly smaller than the number of keywords. Each element vector goes through the transform to obtain its vector of *latent topic weights*. The latter vectors are then compared to each other for similarity. We have used LSI in our prior work [10].

Latent Dirichlet Allocation (LDA). Like LSI, LDA represents all elements as vectors of latent topic weights [4]. The difference is in the transformation from the space of keyword weights to the space of latent topic weights. We have used the open source Java LDA implementation, LDA-J, which is available from the *knowceans.org* site [7]. LDA is implemented with a three-level hierarchical Bayesian model. Each document is modeled as a mixture of k topics for classification. Candidate links between requirements and design elements were recorded for those documents whose Euclidean distance measures were within a given threshold.

---

[1] The method uses the keyword co-occurrence matrix and tries to establish keywords that show "interesting" co-occurrence behavior. This is measured by the χ^2 statistic for each keyword [13].

[2] From the point of view of Information Retrieval, which is used to dealing with millions of documents, all our datasets are small.

[3] In the interest of space, we have relegated the specific formulas we used to estimate these quantities in our work to a tech. report.

3.1. Evaluation measures

Consider a traceability task involving a high-level document consisting of *N* elements and a low-level document consisting of *K* elements. Let the true RTM for this task contain *S* links. Consider an RTM $\mathbb{R}$ containing M < K*N links. Suppose an RTM assessment method F(.) is applied to links in $\mathbb{R}$. Each link *l* from $\mathbb{R}$ can belong to one of the following categories:

- *true positive (hit). l* is in the true RTM and *l* is classified as a true link by F(.).
- *false positive (strike). l* is NOT in the true RTM but *l* is classified as a true link by F(.).
- *true negative. l* is NOT in the true RTM, and *l* is classified as not a link by F(.).
- *false negative (miss). l* is a true link, but it is classified as not a link by F(.).

Let A be the number of *hits*, B be the number of *strikes*, C be the number of *true negatives* and D be the number of *misses*. We know that A+B+C+D = M (only the links in $\mathbb{R}$ are classified by F(.)). We note that F(.) *retains* A+B links and rejects C+D links. We use the following measures in our study.

Recall: $pd = recall = \frac{A}{S}$. Recall is the overall percentage of correct links retained by F(.).

Probability of false positive detection: $pf = \frac{B}{C+B}$.

Pf is the probability of detecting a false positive, i.e., the probability of keeping a non-link in the output. *Pf* represents the probability of false positive detection by method F(.) alone. Additionally, we are interested in the overall probability of detecting a false positive, *opf*, computed as $opf = \frac{B}{(N \cdot K) - S}$.

Precision: $precision = \frac{A}{A+B}$. Precision measures the total percentage of correct links among the links retained by the method.

In our prior studies on trace recovery, we mainly concentrated on *pd (recall)* and *precision* of the recovered candidate RTMs. Our prior results [9,10] show high recall rates in candidate link lists retrieved by individual IR methods. At the same time, in unfiltered candidate link lists, precision was in single digits. The problem of RTM assessment, however, as stated above, is more akin to the traditional classification problem. In addition to *pd*, the probability of false positive detection, *pf*, is traditionally considered as a key measure. In our case, we consider two variations: *pf* and *opf*. The first measure is the probability of false positive detection relative to the false positives contained in the input RTM $\mathbb{R}$. The second measure is the *overall probability of false positive detection* achieved by the combination of the method that produced the candidate RTM $\mathbb{R}$ and our RTM assessment method *F(.)*.

4. Study Design

In this section, we present the datasets used in our studies as well as the design of the studies.

4.1. Datasets Used

In general, the methodology studied here can be applied both to assessment of RTMs between different artifacts and assessment of RTMs between consecutive versions of the same artifact. In this study, our datasets belong to the former class – we had no access to a dataset of the latter category. Both studies described below use data from the CM-1 dataset [14], distributed by NASA's Metrics Data Program (MDP) and available from the PROMISE repository [16]. In our prior work [10], we extracted a requirements document and a design specification from the CM-1 dataset and broke each artifact into individual elements. We also traced the requirements document to the design document via a process described in detail in [10]. For our first study, we used a small subset of the CM-1 dataset. We extracted a single section of the requirements document and the section in the design document to which most of the requirements elements were pointing and restricted our ground truth RTM to the links between the extracted sections. We refer to this smaller dataset as CM-1, subset 1, or as the "student dataset" in the remainder of the paper. Table 1 shows the main characteristics of the main CM-1 dataset and the student dataset.

4.2. Study 1

Our first study used the results of an in-class quasi-experiment, conducted as part of a graduate software engineering topics course in the Spring 2006 semester. 30 students who took the class were divided into two equal size groups. Each student was asked to complete a trace recovery task for the student dataset described above. Students in the first group were asked to perform trace recovery manually, whereas students in the second group were asked to use RETRO [10], a requirements tracing tool built by our group. RETRO was configured to use vector space retrieval with tf-idf weighting schema (see Section 3) to produce candidate link lists.

Table 1. Datasets used in this work.

Dataset:	CM-1	CM-1, subset 1 (student dataset)
# requirements	232	52
# design elements	220	22
# links in RTM	361	35

At the end of the quasi-experiment, 11 submissions from group 1 (manual trace recovery) and 14 submissions from group 2 (trace recovery using RETRO) were deemed acceptable.

We used the committee of all five methods and applied three decision rules: *majority* (3 of 5 methods), *supermajority* (4 of 5 methods), and *consensus* (5 of 5 methods) to classify the links. To find the probability of detection (*pd)* for each committee, we simply ran the true RTM through it. For each of the student-generated RTMs, we therefore were interested in establishing *pf*, the probability of false positive detection, and thus only the false positives from these RTMs were studied. We also looked at how each committee changed the *precision* and *recall* of the RTM.

4.3. Study 2

In the second study, we used the full CM-1 dataset. At the moment, we have not conducted studies in which analysts trace the full CM-1 dataset. In order to evaluate the performance of our assessment techniques on a large dataset, we felt that it was important to test on the full CM-1 dataset. Toward that end, we considered two different ways of automatically seeding (selecting) the RTM with false positive links. In both cases, the false positives were drawn from a list of about 37,000 links – the combined list of links recovered by four (TF-IDF, KE, ProbIR and LSI) of the five methods considered in this study (LDA method did not produce a similarity score that we needed to use in the selection process below). The two methods we considered were:

Random selection. We selected approximately 3% of the links. Each link form the list of about 37,000 links had an equal chance of being selected. We used the "Select a sample" feature of the SPSS statistical package to construct ten datasets.

Weighted selection. Each of the four methods returned a similarity/ranking score for each link, ranging from 0 to 1 (with 1 being most similar and 0 being least similar). We used the sum of these scores as the weight of each link. Using the SPSS "Weight cases by" feature, we weighted our links, and then selected ten 3% samples from the weighted dataset using the "Select a sample" feature of SPSS. Weighted selection is biased *against* our assessment method as false positives with higher rank (and thus – harder to deal with) have a higher probability of being selected.

We used the same voting methods as in Study 1. We examined the input RTMs using the committee of five methods described in Section 3. We used the *Majority*, *SuperMajority*, and *Consensus* decision rules. Finally, we studied both the unfiltered and filtered at 10% inputs from individual methods.

We used the CM-1 answer set to find the recall (*pd*) numbers for each method, and the random and weighted samples to find the probability of false positive detection (*pf*) numbers. For this study, we report mean *pf* values for each selection methods/ voting method combination.

5. Results

Tables 2 through 4 and Figures 1--8 document the results of the first study (the lines connect the results obtained for the same data point on for different decision rules, thus showing the drift of the measures). In Table 2, we show the probability of detection (*pd*, a.k.a., *recall*) rates for different methods for the student dataset used in this study. We show the details for three *committee methods* and our baseline method, *TF-IDF*, which simply looks to see if the line is present in the candidate RTM recovered by the vector space retrieval using *tf-idf* method. In the *unfiltered* case, we used the entire list recovered by all five methods; in the *top 10%* case, we used only the top 10% of these links ranked by similarity, omitting the other 90% of recovered links from consideration. It can be seen, for example, that the majority rule applied to the unfiltered case detects 100% of the correct links as does the TF-IDF method. Tables 3 and 4 show the mean *pf* and *opf* values obtained in our evaluation. Figures 7 and 8 combine the content of Tables 2 and 3 and plot the results of our study in the *pf-vs-pd* space.

Because the candidate RTMs constructed by students from group 1 (manual trace recovery) and those

constructed by students from group 2 (using RETRO) differed significantly, we break our computations by group. In general, we observed that most of the candidate RTMs in group 2 were very similar, and hence we only report the means for this group. Group 1 included 11 different, manually constructed candidate RTMs.

Figures 1 through 6 illustrate the results of applying our RTM assessment methodology to each individual RTM in the group. Figures 1 (no filter) and 2 (top 10% filter) show the *precision-vs-recall* drift, from the original RTM to the RTMs retained by the three committees: *majority*, *supermajority*, and *consensus*. It can be seen, for example, in Figure 1 that Student 6's manually built RTM (original) had recall of 48% and precision of 28%. This drifted to 48%/29% for the Majority rule, to 48% recall/32% precision for the Supermajority rule, and to 38% recall and 41% precision for the Consensus rule.

Table 2. Probability of Detection (pd or recall) rates for CM-1, subset 1 (student) dataset used in Study 1.

	Classifier base	
Rule	*unfiltered*	*top 10%*
Majority	1	0.914286
supermajority	0.971429	0.571429
consensus	0.685714	0.542857
TF-IDF	1	0.571429

Table 3. Means of probability of false positive detection (*pf*) for Study 1 by groups.

		Rule			
Group	**Filter**	*Maj*	*SuperMaj*	*Consensus*	*TF-IDF*
Manual	*None*	0.952	0.817	0.409	0.964
RETRO	*None*	0.915	0.626	0.190	0.967
Manual	*10%*	0.436	0.400	0.187	0.426
RETRO	*10%*	0.097	0.072	0.024	0.085

Figures 3 and 4 show the individual *pf* values for all RTMs and committee methods for filtered and unfiltered cases, respectively. For Student 6's unfiltered RTM, e.g., it can be seen that the Consensus rule had just over a 40% chance of keeping a false positive that it examined (*pf*). Figures 5 and 6 show the *opf* values. It can be seen, e.g., from Figure 5 that the Consensus rule for Student 6's unfiltered RTM had only a 1.8% chance overall of keeping a false positive.

The results of Study 2 are presented in Tables 5 and 6 and in Figure 9. In Table 5, we show the probability of detection (*pd*) rates for the three committee methods for the unfiltered and filtered case (we looked at the Top 10% of the links as in Study 1). It can be seen, e.g., that the Majority rule detected 98% of the correct links in the unfiltered case. Table 6 shows the mean *pf* values obtained in the second study. For example, it can be seen that the Consensus rule only retained 2.9% of the false positives (that were seen) for the randomly selected filtered RTMs.

Table 4. Means of overall probability of false positive detection (*opf*) for Study 1 groups.

Group	**Filter**	Original RTM	Maj	SuperMaj	Consensus	TF-IDF
Manual	None	0.036	0.034	0.029	0.014	0.034
RETRO	none	0.745	0.677	0.463	0.141	0.716
Manual	10%	0.036	0.015	0.014	0.006	0.011
RETRO	10%	0.745	0.072	0.053	0.018	0.063

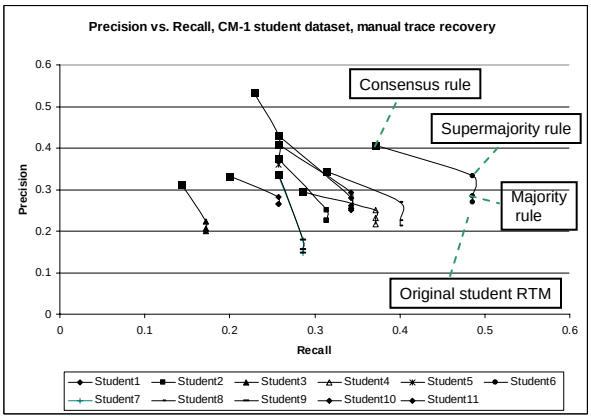

Figure 1. Changes in recall and precision for students from group 1 (manual trace recovery).

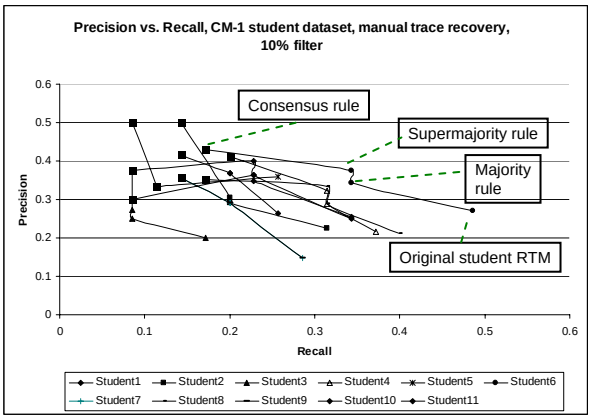

Figure 2. Changes in recall and precision for students from group 1 (manual trace recovery), committee uses only top 10% of links.

Figure 9 shows the mean *pf* versus probability of detection (recall) for the random unfiltered RTMs, the random filtered RTMs, the weighted filtered RTMs, and the weighted unfiltered RTMs. It can be seen, for

example, that the Majority rule had a *pd* of close to 1 and a *pf* of close to 0.8.\

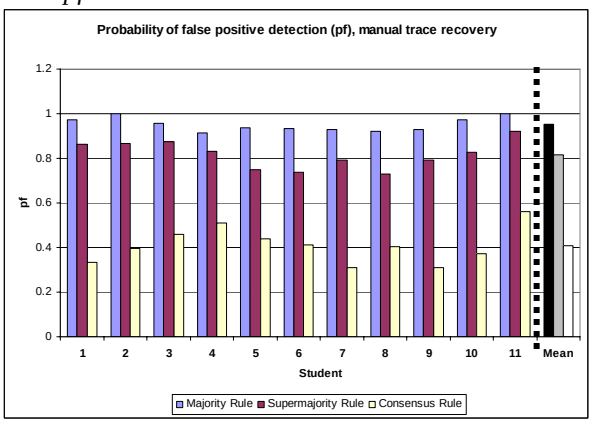

Figure 3. *Pf* values, group 1 (manual trace recovery).

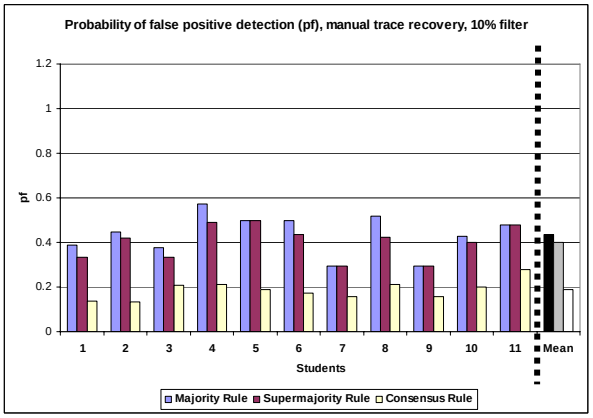

Figure 4. *Pf* values, group 1 (manual trace recovery) committee uses only top 10% of links.

6. Analysis and Discussion

As mentioned previously, our prior work has shown that information retrieval methods perform well in retrieving correct links [10]. Not surprisingly, Study 1 shows that a committee of IR methods that are used to assess a given RTM also does an excellent job of detecting and keeping the true links. Table 2 (*pd* or recall) shows that the Majority rule keeps 100% of the true links in the unfiltered case (tying the TF-IDF method). In the filtered case, the Majority rule performs much better than the TF-IDF method, keeping 91.4% of the true links (as compared to 57.1% for TF-IDF). Even the Supermajority rule ties the TF-IDF method in terms of *pd*. So our prior findings on the effectiveness of information retrieval methods for recovering a high percentage of all true links have also been demonstrated in *assessing* RTMs.

Precision, or the ability to discard false positives, is a noted shortcoming of information retrieval methods [10]. In assessing a given RTM, it is desirable to not only keep all of the correct links evaluated, but also to discard all of the false links evaluated. In data mining, *pf* is traditionally used to measure the probability of false positive detection.

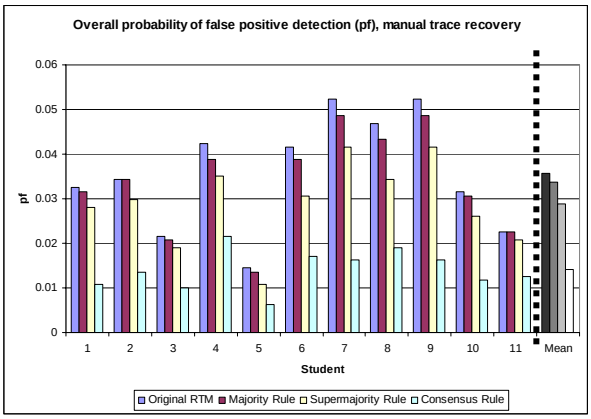

Figure 5. Overall *Pf* values, group 1 (manual trace recovery).

We examine this measure and we also examine the overall *pf* or *opf*. As can be seen in Table 3, the Majority rule, the Supermajority rule, and the TF-IDF method fail to discard 95.2%, 81.7%, and 96.4% (respectively) of the false positives that they examine in the unfiltered Manual (group 1) RTMs. In contrast, the Consensus rule only fails to discard 40.9% of the false positives. This effect is even more prominent for the unfiltered RETRO RTMs where the Consensus rule only fails to discard 19% of the false positives. Filtering has a clear impact on *pf*: the Consensus rule only fails to discard 2.4% of the observed false positives for the filtered RETRO RTMs and all other decision rules perform well. Note that all decision rules outperform TF-IDF except for the Majority rule which only outperforms TF-IDF in the unfiltered case.

Once these false positives have been discarded (presumably they were discarded by the analyst in the RTM building phase), it is of interest to see how well the committees perform at discarding any remaining false positives. Table 4 addresses this question with the *opf* measure. It can be seen that the average original RTM failed to discard 3.6% of the overall false links (in the N x K space) for the manual cases (filtered and unfiltered) and failed to discard 74.5% for both RETRO cases (filtered and unfiltered). In stark contrast, the Consensus rule only failed to discard .06% for the filtered manual group RTMs, 1.4% for the unfiltered manual group RTMs, and 1.8% for the filtered RETRO group RTMs. In other words, the Consensus rule is able to discard almost 100% of the false positives in the N x K space (after an analyst has

discarded anywhere from 8.5% to 96.4% (see Table 3)). Looking at the TF-IDF method, one can see that the Consensus rule committee outperforms TF-IDF in terms of *opf* by 57% (for unfiltered RETRO group RTMs) – that is almost a four-fold improvement.

For Study 2, we focus on *pf* and *pd*. As shown in Table 5, when there is no filtering, the Majority and Supermajority rules capture close to 100% of the true links: 98% and 97% respectively. The Consensus rule retains close to 83% of the true links. In the filtered case, some true links are discarded, with the best performance, 80%, coming from the majority rule.

Table 6 examines the mean probability of false positive detection for the randomly selected RTMs and the weighted selection RTMs. It can be seen that the Consensus rule performs quite well for the filtered RTMs, regardless of the selection method, retaining only 3% of the false links that it examines. On the other hand, the unfiltered cases are a challenge for the committees, with Majority and Supermajority retaining close to 80% of the false positives. The Consensus rule does a better job on the unfiltered case, but still retains almost 40% of the false positives.

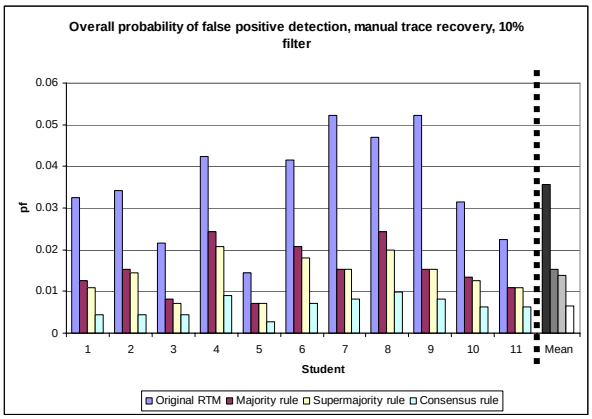

Figure 6. Overall *Pf* values, group 1 (manual trace recovery) committee uses only top 10% of links.

Figure 9 makes it very clear that there is no difference between the selection methods when plotting *pf* vs. *pd*. The random unfiltered and weighted unfiltered values are almost indiscernible. The same is true for the random filtered and weighted filtered cases. It is interesting to note that the Majority rule clearly achieves higher mean *pd*, but at the expense of *pf*. The Consensus rule throws away some true links (hence a decrease in *pd*), but has a very low *pf*. Note that it is desirable to have the *pf* vs. *pd* points in the lower right quadrant of the graph. Our Consensus and Supermajority rules just barely enter this quadrant.

Figures 1 and 2 show a very clear pattern in the behavior of the committee rules. It is apparent in both figures that the original RTM has the highest recall and precision for each student. In Figure 1, we see that the Majority rule increases precision just slightly, with no drop in recall. The Supermajority rule may allow a slight drop in recall, but with an increase in precision. Finally, the Consensus rule improves precision (almost 10%), but at some loss of recall (roughly 10%). In Figure 2, this same trend is apparent, though there is a drop in recall when moving from the original RTM to the Majority rule. As can be expected, the Consensus rule may result in throwing away some true links (since all the methods did not retrieve that true link), but it is usually a small decrease in recall with an increase in precision (as many false positives are discarded).

Figures 3 through 6 demonstrate that, on average, the *pf* is 40% for the Consensus rule applied to the unfiltered manual group RTMs, 80% for the Supermajority rule, and 98% for the Majority rule. *Pf* drops to 19.9% for the filtered manual group RTMs, with the other committees at or just above 40%. This is a significant improvement when using a simple filtering rule (top 10%). The *opf* for the Consensus rule is, on average, only 1.5% for the unfiltered manual case and at/or just above 3% for the other rules and for the original RTM. The *opf* for the filtered case is less than 1% for the Consensus rule, and just around 1.5% for the others. So the Consensus rule is still almost 50% better than the other rules.

6.1 Discussion and Future Work

In our studies we considered two types of candidate RTMs, those created from scratch by human analysts and those built by selecting links from a pool of links retrieved by automated methods. In considering the original problem motivation, it was of great interest whether or not we could use a committee of information retrieval methods to find the mistakes that human analysts introduce when building RTMs. Secondarily, we wondered if a committee of methods could detect problems introduced into an RTM by a single information retrieval method.

Our assessment method captures and rejects a large percentage of the false positives introduced by automated methods used to build the RTM. This is significant because as traceability researchers achieve increased success, more analysts will be using their methods in order to build mappings or RTMs (as illustrated in Section 1). It is thus important that we have techniques available for assessing their accuracy in a manner that is not time consuming or erroneous.

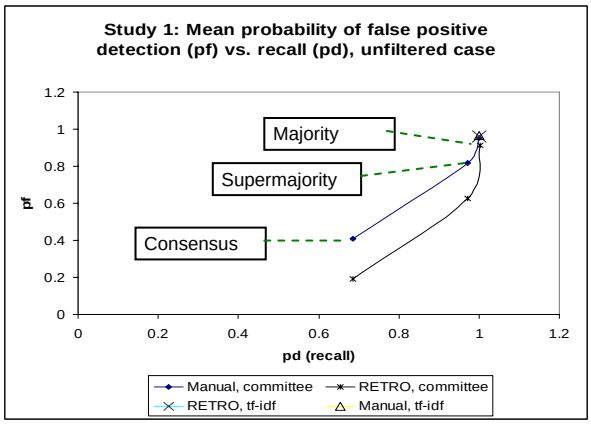

Figure 7. Study 1: Mean probability of false positive detection vs. Recall (unfiltered case).

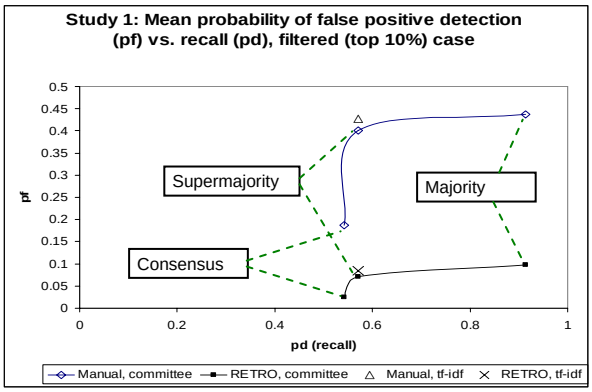

Figure 8. Study 1: Mean probability of false positive detection vs. Recall (filtered case).

Table 5. Study 2: Probabilities of detection (recall)

Filter	Majority	Supermajority	Consensus
None	0.980	0.977	0.828
Top 10%	0.800	0.753	0.518

Perhaps the more interesting aspect of our study is the behavior of our methods on the RTMs that were constructed by the human analysts from scratch. In this particular case, we cannot predict what false positives might be introduced. We see that while our methods were not as good on the human generated RTMs, we still see that our method can reject a significant percentage of the observed false positives while preserving a high *pd*. Humans themselves are good at weeding out false positives (as can be seen by their low *opf* values in Study 1), and our method can come in behind the human analyst and reject a large percentage of the false positives that the human left in.

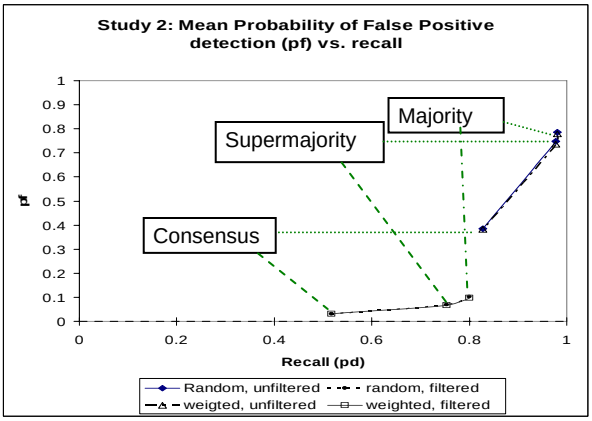

Figure 9. Study 2: Mean probability of false positive detection (*pf*) vs. probability of detection (recall).

Table 6. Study 2: Mean probabilities of false positive detection (*pf*).

		Rule		
Selection Method	Filter	Majority	SuperMajority	Consensus
Random	Top 10%	0.101	0.069	0.029
Weighted	Top 10%	0.096	0.065	0.030
Random	None	0.786	0.748	0.385
Weighted	None	0.779	0.735	0.381

As stated in Section 1, this is our first study of automated methods for RTM assessment. At this point, we see three clear ways in which the approach described in this paper can be improved. First, we can build larger committees by adopting/adapting more IR/text mining methods for trace recovery tasks. Second, each of the methods we used in our committees came with a number of different parameters. Finding the best "variations" for each member method can improve the overall quality of voting. Finally, we have considered a full committee of five methods. However, one can construct five four-member subcommittees and 20 three-member subcommittees just out of the methods used here. Some of these subcommittees might prove to be more sensitive to false positives than the overall committee. To show this, consider the information in Table 7. There, we report pair-wise Pearson correlations between the lists of false positive links in the *filtered* committees used in our Study 2. As seen from this data, TF-IDF and KE methods show a very high correlation and, therefore, provide almost no *orthogonal* information --- on the contrary, these methods *reinforce each others' false positives during "committee hearings"* This correlation was not unexpected: as we state in Section 3, the KE method operates on the data generated by the TF-IDF method.

A more pleasant surprise to us was relatively low correlation rates for other methods. At the same time, we note that other methods show a much more infrequent rate of false positive co-incidence which suggests that careful selection of subcommittees may improve the false positive rejection rates without affecting recall.

Table 7. Pearson correlations between false positives in traces recovered by five methods for CM-1 dataset, filtered at top 10%.

	TFIDF	KE	ProbIR	LSI	LDA
TFIDF					
KE	0.932				
ProbIR	0.210	0.209			
LSI	0.499	0.485	0.170		
LDA	-0.44	-0.41	0.176	-0.30	

7. Acknowledgments

Our thanks to Stephanie Ferguson and Marcus Fisher. We thank Mike Chapman and the Metrics Data Program (MDP) for access to the CM-1 dataset. This work is sponsored by NASA under grant NAG5-11732.

8. References

[1] Antoniol, G., , G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Recovering Traceability Links between Code and Documentation. *IEEE Transactions on Software Engineering*, Volume 28, No. 10, October 2002, 970-983.

[2] Antoniol, G., Merlo, E., Guéhéneuc, Y., and Sahraoui, H. 2005. On feature traceability in object oriented programs. in *Proc. 3rd Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE'05)* 2005, 73-78.

[3] Baeza-Yates, Ricardo A., Berthier A. Ribeiro-Neto: Modern Information Retrieval. ACM Press / Addison-Wesley, 1999.

[4] Blei, D., A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.

[5] Brilliant, S. S., Knight, J. C., and Leveson, N. G. 1990. Analysis of Faults in an N-Version Software Experiment,. *IEEE Trans. Softw. Eng.* 16, 2 (Feb. 1990), 238-247.

[6] Deerwester, S., S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman, Indexing by Latent Semantic Analysis, *J. Am. Soc. Information Science*, vol. 41, no. 6, pp. 391-407, 1990.

[7] Heinrich, Gregor, "LDA-J Library" Code available at http://www.arbylon.net/projects/.

[8] Cleland-Huang,J., C.K. Chang, G. Sethi, K. Javvaji, H. Hu, and J Xia. Automating speculative queries through event-based requirements traceability. *Proc. International Requirements Engineering Conference (RE'02)*, 2002.

[9] Huffman Hayes, Jane, Alexander Dekhtyar, Senthil Sundaram, Sarah Howard, "Helping Analysts Trace Requirements: An Objective Look," in Proceedings of IEEE Requirements Engineering Conference (RE) 2004, Kyoto, Japan, September 2004, pp. 249-261.

[10] Huffman Hayes, J.,. Dekhtyar, A., and Sundaram, S.. Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods *IEEE Transactions on Software Engineering*, Volume 32, No. 1, (January 2006), 4-19.

[11] Huffman Hayes, J., Dekhtyar, A., Holbrook, E.A., Sundaram, S., Dekhtyar, O., Will Johnny/Joanie Make a Good Software Engineer?: Are Course Grades Showing the Whole Picture?, in *Proc., Conference on Software Engineering Education and Training (CSEET)*, 2006, pp. 175 - 182.

[12] Marcus, A., and J. Maletic, Recovering Documentation-to-Source Code Traceability Links using Latent Semantic Indexing, *Proc. ICSE* 2003, pp. 125 – 135.

[13] Matsuo, Yutaka, and Mitsuru Ishizuka: Keyword Extraction from a Single Document using Word Co-occurrence Statistical Information. *FLAIRS Conference* 2003: 392-396.

[14] MDP Website, CM-1 Project, http://mdp.ivv.nasa.gov/mdp_glossary.html#CM1, 2005.

[15] Poshyvanyk, D., Gueheneuc, Y., Marcus, A., Antoniol, G., and Rajlich, V. Combining Probabilistic Ranking and Latent Semantic Indexing for Feature Identification in *Proc.. ICPC* 2006, pp. 137-148.

[16] PROMISE Repository, http://promise.site.uottawa.ca/SERepository/.

[17] J. H. Hayes, A. Dekhtyar, S. Sundaram, Text Mining for Software Engineering: How Analyst Feedback Impacts Final Results, in *Proc. MSR'2005: Second International Workshop on Mining Software Repositories*, pp. 58-62, St. Louis, MO, May 2005.