Towards New Ways of Evaluating Methods of Supporting Requirements Management and Traceability using Signal-to-Noise Ratio

Krzysztof Wnuk[1], Markus Borg[2] and Tony Gorschek[1]
[1]*Software Engineering Department, Blekinge Institute of Technology, Karlskrona, Sweden*
[2]*Software and Systems Engineering, Laboratory at RISE Research Institutes of Sweden, Sweden*

Keywords: Requirements Management, Traceability, Information Retrieval, Information Distance.

Abstract: Developing contemporary software solutions requires many processes and people working in synergy to achieve a common goal. Any misalignment between parts of the software production cycle can severely impede the quality of the development process and its resulting products. In this paper, we focus on improving means for measuring the quality of methods used to support finding similarities between software product artifacts, especially requirements. We propose a new set of measures, Signal-to-Noise ratios which extends the commonly used precision and recall measures. We test the applicability of all three types of SNR on two methods for finding similar requirements: the normalized compression distance (NCD) originating from the domain of information theory, and the Vector Space Model originating from computer linguistics. The results obtained present an interesting property of all types of SNR, all the values are centered around 1 which confirms our hypothesis that the analyzed methods can only limit the search space for the analysis. The analyst may still have difficulties in manually assessing the correct links among the incorrect ones.

1 INTRODUCTION

The size of software system continues to grow not only in terms of code lines, but also in terms of its augmenting factors such as complexity, the degree of heterogeneity and decentralization. These augmenting factors are recognized as the "hot spots" research topics in Requirements Engineering (RE) (Cheng and Atlee, 2007). This problem is not new, since many studies reported a need for revisiting current software and requirements engineering techniques under pressure of growing size and complexity (Bergman et al., 2002; Finkelstein, 1994; Leveson, 1997; Northrop et al., 2006; Maccari, 1999).

Providing automatic support for the most time-consuming activities of requirements engineering and management has been recognized as one of the challenges in development and large and complex systems (Konrad and Gall, 2008; Berenbach et al., 2009; Finkelstein, 1994). In particular, techniques and tools that can ease, and partially automate the task of identifying and document traceability links among requirements artifacts and between requirements and other artifacts (Cheng and Atlee, 2007; Cleland-Huang et al., 2004; Hayes et al., 2006a; Marcus and Maletic, 2003; Sabetzadeh and Easterbrook, 2005; Cleland-Huang et al., 2005; Karlsson et al., 2007).

Several methods have been proposed and evaluated for semi-automated tracing requirements, including scenario and test case-based methods (Egyed, 2003), policy-based methods (Murta et al., 2006), event-based methods (Cleland-Huang et al., 2003a) and rule-based approaches (Spanoudakis et al., 2004). Three IR-rooted methods dominate semi-automatic and automatic support for requirements traceability: Latent Semantic Analysis (LSA), the Vector Space Model, and probabilistic approaches (Hayes et al., 2006a; Hayes et al., 2008; och Dag et al., 2004; Deursen et al., 2006; De Lucia et al., 2004; Antoniol et al., 2000).

Despite delivering promising results in terms of precision and recall, the methods have three major pitfalls: (1) they require pre-processing of data, sometimes manually, (2) their performance heavily depends on the data input and its quality and (3) they leave the analysts with a list of candidate link that he has to manually investigate. Thus, these semi-automatic methods only help to effectively reduce the search space rather than making actual decisions.

In that case, using precision and recall to evaluate the mentioned methods can be questioned in the following way: are these quality measures providing

Wnuk, K., Borg, M. and Gorschek, T.
Towards New Ways of Evaluating Methods of Supporting Requirements Management and Traceability using Signal-to-Noise Ratio.
DOI: 10.5220/0007717203300339
In *Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2019)*, pages 330-339
ISBN: 978-989-758-375-9

the full picture of the support of the linking process or maybe only precisely describe how well the search space is limited? One of the studies that uses machine learning approach for tracing regulatory codes to product specific requirements reported that one of the methods used in the study successfully excluded 1806 of the potential 1889 links. (Cleland-Huang et al., 2010). This, however, means that 83 links were left for the analyst to analyze and the precision, in this case, was below 10%! How many links are presented for the analysts: hundreds or maybe only ten or twenty? How easy it is for the analyst to detect "false positives"? How the presence of false positives next to correct answer impact the judgment of the requirements analyst?

As identified by (Järvelin and Kekäläinen, 2002), it is necessary to develop measures, going beyond precision and recall, which credit methods that clearly distinguish between highly relevant and less relevant. Without good measures to base evaluations on, future improvements to traceability tools are hard to make in a systematic way.

In this paper we propose extending the current way of accessing performance and quality of semi-automatic methods for supporting requirements traceability. We propose using the Signal-to-Noise Ratios (SNR) as a complementary quality measure that can bring more qualitative insights into the performance of tested methods. We complement other research efforts but expanding the already presented measures, e.g., Mean Reciprocal Rank, Accuracy@N, Recall-Rate@N, Mean Average Precision (Zhou et al., 2012).

We discovered that most values are centered around 1 which may cause difficulties in manually assessing the correct links among the incorrect ones. We applied SNR to the new methods for measuring similarity between requirements that does not require any pre-processing, based on Normalized Compression Distance (NCD) from the information theory domain (Vitányi et al., 2008).

This paper is organized as follows: Section 2 describes related work and bring some rationale into research. Section 3 gives the theoretical background of SNR and Section 4 proposes how SNR can be used in software engineering. Section 5 presents results from measuring SNR for two methods for supporting the requirements consolidation task. Section 6 presents the results of our case study and our interpretations of the data, Section 7 discusses threats to validity and in section 8 we draw conclusions and directions for future work.

2 RELATED WORK

The key activities of requirements traceability are establishing and maintaining traceability links between artifacts. Many of the current state-of-the-art tools support semi-automatic link generation or identification. In most cases, a human analyst is presented a list of possible candidates, thus significantly reducing the search space, but the actual decision making must be done manually. The analyst can actually discard true links and make the results of semi-automated traceability worse (Hayes et al., 2005).

The main purpose of Information Retrieval (IR) methods is to extract relevant items and at the same time retrieve as few of the non-relevant as possible. The most relevant candidates should be ranked first when presented to the user, in order to support relevance judgments and decision making.

Various similarity techniques have been used to support traceability in the software engineering domain. Natt och Dag et al. used vector-space models to measure linguistic similarities to link market requirements to product requirements using the tool ReqSimile (och Dag et al., 2005). Hayes et al. developed the tool RETRO to link requirements to design documents and bug reports, by default also using vector-space models (Hayes et al., 2008). Latent Semantic I Indexing (LSI) is an extended vector-space based method that has been used (Marcus and Maletic, 2003; De Lucia et al., 2008), and probabilistic methods like what is used in spam filters has also been tried (Antoniol et al., 2000). Cleland-Huang et al. proposed a new method of traceability based upon event-notification applicable even in a heterogeneous and globally distributed development environment (Cleland-Huang et al., 2003b). Huffman Hayes et al. proposed the RETRO tool for improving the overall quality of the dynamic candidate link generation for the requirements tracing process (Hayes et al., 2006b).

Even though these IR techniques indicate usefulness, in all cases there is considerable scope for improvements. A study conducted by Lee et al., claims that assessment of semantic similarity fundamentally is a human cognitive capability, and neither of the vector-space oriented techniques word-based, n-gram nor latent semantic analysis produced good correlations with human judgments (Lee et al., 2007). All approaches mentioned, including direct spam filter evaluations as (Tariq Banday and Jan, 2009), could utilize a measure for decision making. Hayes et al. stressed that the issue of analyst interaction with software needs further studies (Hayes et al., 2006b).

Hayes et al. (Hayes et al., 2003) present re-

Table 1: The results from algorithms after trimming (Hayes et al., 2003). R indicates Recall and P precision.

	Vanilla algorithm (10x10)	Retrieval with key phrases algorithm (10x10)	Retrieval with thesaurus algorithm (19x50)
Top 4	R: 23% P: 17.6%	R: 27.2% P: 5.2%	R: 85.4% P: 40%
Above 25	R: 23% P: 75%	R: 27% P: 25%	R: 9.7% P: 40%
Within 33	R: 23% P: 23%	R 27.2% P: 15.7%	R: 48.7% P: 44.4%
Within 50	R: 33% P: 20%	R: 27.2% P: 15.7%	R 58.5% P: 42.1%

sults in terms of precision and recall depending on the threshold given in number of candidate requirements showed by the algorithms. They decided to decrease the size of the list in order to improve precision as a potential cost to recall. The selected values of the threshold were: top 4 candidates, or candidates with similarity above 0.25, any candidates with a similarity within 0.33 and 0.50 of the similarity of the top candidate. The results do not reveal any particular function or relationship that can be later explored. For example for the vanilla algorithm, both above 25 and within 33 returned the same recall 23%, but above 25 returned precision of 75% while within 33 only 23%.

Several measures have been proposed to better evaluate IR methods. Järvelin and Kekäläinen (Järvelin and Kekäläinen, 2002) propose measures based on cumulated gain, combining the degree of relevance and rank. This gives an indication about the quality of a method, but does not evaluate the support for decision making. Spink and Greisdorf (Spink and Greisdorf, 2001) suggest the median effect as a measure to evaluate the way the distribution of relevance judgments of retrieved items are generated by a method, but this is mainly focused on being an alternative to dichotomous measures. Kekäläinen and Järvelin (Kekäläinen and Järvelin, 2002) also, identify the weakness of just evaluating binary relevance as is the case for recall and precision, and they propose generalized recall and precision, which reward IR methods that extract more relevant items. These measures also do not evaluate how easy it is to make decisions. While Zhou et al. focused on measures that can evaluate a process that produces a list of possible responses to a query, e.g. Mean Reciprocal Rank, Mean Average Precision (Zhou et al., 2012; Manning et al., 2008), the SNR introduced in this work tries to describe how much the analyst is distracted by the presence of false positives next to a correct answer in the candidate list.

Other measures as expected search length (Cooper, 1968), normalized recall measure (Rocchio, 1966), sliding ratio measure (Pollack, 1968), satisfaction-frustration-total measure (Myaeng and Korfhage, 1990) can all be used to credit methods presenting relevant items high up the list, but again evaluation of decisions making support is not targeted.

3 DEFINITIONS AND THEORY

The signal-to-noise ratio (SNR) is a measure widely used in science and engineering to quantify how much a signal has been corrupted by noise. It compares the level of a desired signal to the level of background noise which is a fundamental measure in signal processing theory.

$$SNR = \frac{P_{signal}}{P_{noise}} \tag{1}$$

$$SNR = \frac{mu}{sigma} \tag{2}$$

In its classical definition, signal-to-noise is a ratio between the average power of the signal (P_{signal}) and the average power of the noise (P_{noise}) (1) (Gonzalez and Woods, 2006). Since signals have usually a very wide dynamic range, the logarithmic decibel scale is used to express SNR. There exist alternative definitions of SNR, like for example the ratio of mean (mu) to standard deviation (sigma) of a signal or measurement (2). This definition is commonly used in image processing (Gonzalez and Woods, 2006; Stathaki, 2008; Raol, 2009; Russ, 1999) where the SNR of an image is usually calculated as the ratio of the mean pixel value to the standard deviation of the pixel values over a given neighborhood.

The higher the ratio, the less obtrusive the background noise is. In other words it is a ratio between the meaningful information and the background noise. In our case, the signal is represented by the correct link between two objects while the noise is represented by all other potential links. In digital signal processing the noise is the error signal cause by the quantization of the signal, assuming that the analog-digital conversion has been performed.

$$F = 2 * \frac{precision * recall}{precision + recall} \tag{3}$$

$$Precision = \frac{true\_positives}{true\_positives + false\_positives} \quad (4)$$

$$Recall = \frac{true\_positives}{true\_positives + false\_negatives} \quad (5)$$

Precision and recall are widely used for statistical classifications. They can be seen as measures of exactness or fidelity (precision) or completeness (recall). The relationship between precision and recall is often inverse, the effort put on increasing the recall of the method often result in a decrease of precision and vice-versa. Therefore, these measures should not be discussed in isolation and are often combined into the single measure, such as the F-measure, which is the weighted harmonic mean of precision and recall (see equation 3 for an evenly weighted version of F measure). Finally, based on the notion of true positives, true negative, false positives, and false negatives we can classify the results of the information retrieval task and build additional measures.

For example, recall can be called true positive rate 4, or true negative rate can be called specificity 5. These measures can be suitable for measuring the quality of the result of the automatic classification task, but do not assume that someone, for example a requirements analyst will use the results as a list of possible candidates in linking similar objects (Natt och Dag et al., 2006). In this case, the measures described above give only an indication of the entire result set, without assessing the quality of how distinctive the correct answers are from the incorrect ones. We assume that this support can help the human analyst to assign more correct links and minimize the number of false positive links (Natt och Dag et al., 2006).

4 RATIONALE FOR DEFINING SIGNAL-TO-NOISE RATIO

To illustrate the rationale for defining yet another measure we use an example of linguistic tool support for requirements consolidation (Natt och Dag et al., 2006). In this case, a similarity measure has been utilized to propose a list of requirements candidates and their similarity score to one actually analyzed. The requirements analyst uses the resulting list of candidates to assign links between two or more similar requirements. In the experiment performed to validate the method (Natt och Dag et al., 2006), a set of 30 requirements has been analyzed against 160 other requirements. The key with correct answers, consisting of 20 links, had been prepared before the experiment. Once the correct links have been assigned, they have been checked with the output of the tool to assess which position on the candidate list they will be classified.

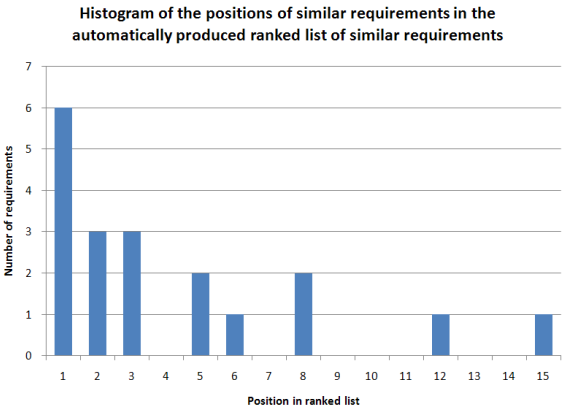

Figure 1: Histogram of the positions of similar requirements in the automatically produced ranked list of similar requirements (Natt och Dag et al., 2006).

Figure 1 depicts a histogram of the distribution of the positions at which the correctly similar requirements end up in the ranked list produced by the tool (Natt och Dag et al., 2006). For the data set used in (Natt och Dag et al., 2006), almost all (17 out of 20) of the correct answers end up at position 8 or better in the ranked list, and could therefore be quickly spotted. The question that remains unanswered here is what are the similarity scores for all the incorrect links proposed on the list. In case the correct answer has position 8 on the list, there seems to be 7 other answers ranked as more similar than the correct answer, which the analyst has to correctly disregard in order to create the correct link. The decision which links to reject may be hard if the similarity scores of all incorrect but highly ranked requirements are very similar to the score of the correct answer. In other words, the analysts may get mislead by the fact that the *noise* is highly ranked than the *correct signal*. Thus, we propose applying three SNR measures that gives insights of the differences between the correct answer and the very close appearing incorrect answer.

To address the issues mentioned above, we propose three versions of signal-to-noise that can be applied for evaluation of automatic methods for supporting decision-oriented tasks in software engineering. The first measure, called *Individual SNR* is the ratio between the correct answer and the closest incorrect answer on the list of candidates. In a case when the correct answer is not ranked as number one on the list of candidates, the *Individual SNR* is an average value of the two closest incorrect proposals, one higher than the correct one and one lower than the correct answer 6. The interpretation is a type of SNR may be that

it is the ability to spot the correct answer among the closest proposed incorrect answer(s).

$$SNR_{individual} = \frac{score\_corr}{(score\_incorr\_down + scope\_incorr\_up)/2} \quad (6)$$

The second version of SNR we propose is the *Maximum Noise SNR* which is the ratio of the similarity degree of the correct answer of the candidates list to the maximum value of similarity of the incorrect answer presented in the list of candidates 7.

$$SNR_{Max} = \frac{score\_corr}{\sum_{MAX} score\_incorrect} \quad (7)$$

The third version of SNR we propose in this paper, called *Average SNR*, is the ratio between the value of similarity of the correct answer to the average noise ratio value above a certain threshold 8. The threshold in Figure 1 has been set to 15, which means that candidate requirements that scored below number 15 on the list were not considered. However, this is a modified version of the *Average SNR* measure. Our definition consider a value rather than a certain number on the ranked list as the threshold.

$$SNR_{Avg} = \frac{score\_corr}{\sum_{i=1}^{n} \frac{score\_incorrect}{nbr\_of\_incorrect_{h}its}} \forall i < threshold \quad (8)$$

5 MEASURING SNR FOR THREE REQUIREMENTS SIMILARITY METHODS

In this section, we present results from measuring the three versions of SNR on the set of requirements described in section 4. The requirements are reused from the experiment on a linguistic tool, supporting the consolidation of requirement sets (och Dag et al., 2005). The analysis has been performed for two methods of measuring the similarity between the requirements: (1) the linguistic method based on vector space model (och Dag et al., 2005; Natt och Dag, 2006b) and (2) the normalized compression distance method based on information theory (Cilibrasi et al.,) and POIROT (Lin et al., 2006).

5.1 Data in the Context

The set contains 30 requirements that are checked against 160 requirements. There exist 20 correct links between the requirement sets. Knowing which links are correct, we have compared the values of the similarity score for the correct and incorrect answers.

Next, we used the ReqSimile tool (och Dag et al., 2005; Natt och Dag, 2006b), which implements the linguistic method, and the Complearn implementation of normalized compression distance (Cilibrasi et al.,) to measure the SNR values for the two techniques. The linguistic method uses a vector-space representation of requirements where each requirement is represented using a vector of terms with a respective number of occurrences (och Dag et al., 2004; Manning and Schütze, 2002). From the matrix which shows how many times a term appears in each requirement the information may be derived about how many terms the two requirements have in common i.e. overlap. The very similar requirements will result in closely clustered points in this vector space (Manning and Schütze, 2002). In the evaluated method (och Dag et al., 2005) a frequency of terms has been used, instead of counting the occurrences. The cosine correlation measure is often chosen in text retrieval applications for the purpose of finding similar requirements, as it does not depend on the relative size of the input (Manning and Schütze, 2002).

$$\sigma(f,g) = \frac{\sum_{t} w_f(t) * w_g(t)}{\sqrt{\sum_{t} w_f(t)^2 * \sum_{t} w_g(t)^2}} \quad (9)$$

The measure in 9 is used, where f and g are two requirements, t ranges over terms, and $w(t)$ denotes the weight of term t. The term weight is typically a function of the term frequency, since while the number of times a word occurs is relevant, its relevance decreases as the number gets larger (Manning and Schütze, 2002). As mentioned in (Natt och Dag, 2006a), there is no guarantee that two requirements that are similar according to the $\sigma(.)$ measure are indeed related. The method evaluated does not consider hypernyms and hyponyms (Jackson and Moulinier, 2002). POIROT is a Web-based tool supporting traceability of distributed heterogeneous software artifacts. A probabilistic network model is used to generate traces between requirements, design elements, code and other artifacts stored in distributed 3rd party case tools such as DOORS and source code repositories (Lin et al., 2006).

Procedures of the Case Study. To avoid errors while measuring, the analysis has been performed

twice by two researchers, working independently and then compared. Any inconsistencies and possible errors were discussed and corrected.

5.2 Results of Measuring SNR on the First Dataset

The results from measuring the three types of SNR are presented in Tables 2, 3, and 4. For the average SNR calculations we use similarity 0.5 as the threshold in the linguistic approach. The very low similarity measures of the NCD forced us to lower the threshold to 0.4 for this method, accepting that direct comparisons no longer are possible. R19, the only quality requirement, was in all calculations considered as an outlier and disregarded. Its format and structure lead to too high similarity values.

A quick comparison between the three techniques can be seen in Table 5. The SNR values are in all cases close to 1 and the correct link is in the most cases not the first on the list. The linguistic VSM based approach has in all cases higher SNR than the proposed method from the domain of information theory. POIROT seems to deliver promising values for some requirements R1 (SC13) and R2 (41104).

As it can be seen in Table 2 only for one requirement (R19) the SNR value is more than 2. In all other cases the values ranges from 0.69 to 1.2 for various types of SNR. The *Na* values in Table 2 for the AvgSNR corresponds in this case to the situation when the measurement could not be performed: for example in cases of R9 and R16 both the correct answer and the highest noise were below threshold 0.5. Moreover, for some data points, the values for MaxSNR and AvgSNR are identical (R3 and R17), which means in this case that there are only two data points above the threshold and the top candidate is the noise (R3 case) or a correct signal (R17). The equal values for both IndSNR and MaxSNR indicate that the correct answer is number one on the list of candidates (R2, R5, R11, R17, R18, and R19). The average value for IndSNR is slightly higher (1.035) than for MaxSNR (0.93) and AvgSNR (0.974) 5. Finally, the biggest median value represents results for AvgSNR (1.06), comparying to IndSNR(1.009) and MaxSNR(0.94)

The results for measuring all three types of SNR for the Normalized Compression Distance method of assessing the similarity between requirements are more dispersed, see Table 3. The values range from 0.28 to 1.04. The correct answer mostly ended up on the list of candidates much lower than for the linguistic similarity measure method (only for 4 cases it is within the top 10 candidates). The average for IndSNR is the highest (1), with more differences to other types of SNR (MaxSNR average is equal to 0.72 and AvgSNR average is equal to 0.759 in this case). The similar, bigger than in the case of linguistic support, difference can be seen between the medians for IndSNR (1), MaxSNR (0.75) and AvgSNR (0.75) respectively.

Table 2: The results from measuring all three types of SNR for the ReqSimile tool that uses vector space model.

Requirement	IndSNR	MaxSNR	AvgSNR	Pos. on the list
R1 (SC13)	1.04	0.98	1.16	2
R2 (41104)	1.26	1.26	1.34	1
R3 (41112)	1.12	0.97	0.97	2
R4 (41114)	1	0.68	0.68	3
R5 (41123)	1.04	1.04	1.14	1
R6 (41301)	1.005	0.89	0.95	5
R7 (41307)	1.009	0.57	0.62	64
R8 (41309)	0.97	0.79	0.83	15
R9 (41414)	1.02	0.91	Na	3
R10 (41601)	0.99	0.95	1.19	3
R11 (41606)	1.008	1.008	1.07	1
R12 (41608)	0.99	0.94	1.06	5
R13 (41710)	1.02	0.89	0.92	8
R14 (41804)	1.009	0.94	1.20	6
	1.001	0.91	1.20	8
R15 (41811)	1.001	0.89	1.06	12
R16 (4205)	1.01	0.89	Na	2
R17 (43302)	1.12	1.12	1.12	1
R18 (43303)	1.05	1.05	1.05	1
R19 (43402)	2.7	2.7	Na	1

The result for the POIROT tool (see Table 4 are similar than those for ReqSimile. POIROT returns much better IndSNR for R1 and R2 and similar values for the remaining requirements.

6 CASE STUDY RESULTS AND INTERPRETATION

As it can be seen from the results depicted in Tables 2, 3 and 4 the values for all three types of SNR are centered around 1. This may lead to the following

Table 3: The results from measuring all three types of SNR for the tool using NCD.

Requirement	IndSNR	MaxSNR	AvgSNR	Pos. on the list
R1 (SC13)	1.03	0.93	0.97	4
R2 (41104)	1	0.49	0.49	111
R3 (41112)	1	0.75	Na	20
R4 (41114)	1.06	0.84	Na	5
R5 (41123)	1.01	1.01	1.04	1
R6 (41301)	0.99	0.42	0.48	155
R7 (41307)	1.003	0.66	0.72	97
R8 (41309)	0.998	0.84	0.86	57
R9 (41414)	1	0.597	0.62	134
R10 (41601)	1.007	0.66	0.79	80
R11 (41606)	1.01	0.8	Na	52
R12 (41608)	0.98	0.85	0.92	13
R13 (41710)	1	0.89	1	9
R14 (41804)	0.993	0.91	0.99	14
	1	0.77	0.84	73
R15 (41811)	1	0.73	NaN	95
R16 (4205)	0.93	0.507	0.57	92
R17 (43302)	0.99	0.564	0.564	41
R18 (43303)	1	0.537	0.537	29
R19 (43402)	0.996	0.28	0.389	136

Table 4: The results from measuring all three types of SNR for the POIROT tool. In this case we calculated average SNR for all data point with confidence level >=50%.

Requirement	IndSNR	MaxSNR	AvgSNR	Pos. on the list
R1 (SC13)	1.31	0.775	2.05	3
R2 (41104)	2.01	2.01	2.56	1
R3 (41112)	1	0.766	0.918	12
R4 (41114)	1.02	0.75	1.19	4
R5 (41123)	1.18	0.99	1.92	2
R6 (41301)	1	0.76	0.60	43
R7 (41307)	0.99	0.83	0.83	28
R8 (41309)	1	0.71	0.53	47
R9 (41414)	0.98	0.93	1.38	3
R10 (41601)	0.97	0.92	1.25	2
R11 (41606)	0.98	0.775	1.14	4
R12 (41608)	0.98	0.85	1.01	3
R13 (41710)	1.01	0.82	0.96	11
R14 (41804)	1	0.89	0.97	11
	1	0.85	0.86	18
R15 (41811)	1	0.79	0.89	16
R16 (4205)	0.995	0.4	0.19	34
R17 (43302)	1.3	1.3	1.94	1
R18 (43303)	1.06	1.06	2.5	1
R19 (43402)	3.27	3.27	5.88	1

interpretation: in the task of requirements consolidation for the methods that have been tested, the differences between the values of similarity of the correct answer and the incorrect answers are very small. Thus, SNR offers better understanding and new insights about these methods. Still, we should further investigate how the human analyst reacts to a list of X candidates with similar SNR values. Paradoxically, the struggle to improve IR-methods may result in a candidate list of very similar requirements and those make the final decision more difficult.

The results also provide valuable information about the nature of candidate links that an analyst has to examine while assigning links. The low levels of SNR obtained in this study suggest that the analyst may have difficulties assessing which of the proposed candidates is the correct one, especially if the differences of the degree of similarity are very small. As a result, in this case, the analysts must turn back to the original text to make the judgment, or more or less guess the correct answer by selecting a requirement from the list of candidates.

7 THREATS TO VALIDITY

We discuss validity threats based on the classification of threats provided by Yin (Yin, 2002). To address *reliability* we have measured all three types of SNR for three methods that can be used to help finding similar requirements. Moreover, the data set selected for this study is reused from a previous study on supporting similarity analysis of requirements from multiple sources together with the correct answer. Finally, both requirements sets and correct answer can be shared upon a request for replications.

Due to the descriptive and exploratory nature of

Table 5: Comparison of the three SNR measures and position of the correct link for Reqsimile,NCD and POIROT.

	ReqSimile	NCD	POIROT
Average IndSNR	1.035	1	1.20
Median IndSNR	1.009	1	1
Average MaxSNR	0.930	0.724	1.02
Median MaxSNR	0.94	0.75	0.84
Average AvgSNR	0.974	0.759	1.48
Median AvgSNR	1.06	0.75	1.07
Average Position	7.526	56.947	12.25
Median Position	3	52	4

this study, where the main focus is to introduce a new set of measures and to provide an example without an attempt to draw any causal relationship, the internal threats to validity are minimized. Still, it would be valuable to perform a user study where we could ask the participants to perform a task and see if SNR results "matches" the performance of users in performing this tasks (Kochhar et al., 2016). The assumed positive causal relationship of SNR introduction should be confirmed in user surveys.

Since this study proposes only one way of measuring the differences between the candidate links presented to the analysts, the threats to construct validity remains not fully addressed. It remains to be an open question how SNR performs in relation to other metrics suggested to evaluate IR techniques applied to software engineering, e.g., Mean Reciprocal Rank, Accuracy@N, Recall-Rate@N, Mean Average Precision, just to name a few.

Finally, the threats to external validity have been partially addressed by measuring the three types of SNR for two methods that may be used as support for linking similar requirements. Moreover, there has been several new IR-based methods proposed. Measuring SNR on as many IR-based methods as possible remains to be the part of future work. Moreover, We plan to expand our analysis to other traceability recovery techniques, e.g., bug localization (Rath et al., 2018). The used dataset is small and we plan to replicate our study on much larger datasets.

8 CONCLUSIONS AND FUTURE WORK

Producing software is a complex task and keeping the involved processes aligned is an important challenge (Sabaliauskaite et al., 2010). One approach to improving the alignment is to utilize traceability techniques to create and maintain links between software artifacts. Most of the traceability techniques currently available are semi-automatic (Raja and Kamran, 2008). They present a list of candidates and a human analyst has to make the final decision, whether a link should be established or not. The commonly accepted measures for evaluating traceability techniques fail short on measuring the support for this decision.

In this paper, we propose a new way of measuring the quality of the list of candidates for semi-automatic methods. We apply a new set of measures, signal-to-noise ratio, to a set of requirements and evaluate three methods. The results show that the differences between the signal and noise are small. This clearly indicates that an analyst may have a problem when making the decision. We propose that the measures should be used to help improve the semi-automatic techniques, and also to lay the foundation for more automated tools in the future. Another contribution of this paper is that we have tried normalized compression distance as a method to assess the similarity between software artifacts, even though the results are discouraging.

Future work includes expanding the data set with more textual requirements and test cases, and measuring SNR for more of the available techniques. We also plan to further explore the suitability of NCD for non-textual requirements tracing.

ACKNOWLEDGMENTS

This work is supported by the Knowledge Foundtion in Sweden within the Software Engineerng Rethought project, https://rethought.se/.

REFERENCES

Antoniol, G., Canfora, G., de Lucia, A., and Casazza, G. (2000). Information retrieval models for recovering traceability links between code and documentation. *Software Maintenance, IEEE International Conference on*, 0:40.

Berenbach, B., Paulish, D. J., Kazmeier, J., and Rudorfer, A. (2009). *Software & Systems Requirements Engineering: In Practice*. Pearson.

Bergman, M., King, J. L., and Lyytinen, K. (2002). Large-scale requirements analysis revisited: The need for understanding the political ecology of requirements engineering. *Req Eng*, 7(3):152–171.

Cheng, B. H. C. and Atlee, J. M. (2007). Research directions in requirements engineering. In *FOSE '07: 2007 Future of Software Engineering*, pages 285–303, Washington, DC, USA. IEEE Computer Society.

Cilibrasi, R., Cruz, A. L., de Rooij, S., and Keijzer, M. The complearn suite website. http://www.complearn.org/index.html.

Cleland-Huang, J., Chang, C. K., and Christensen, M. (2003a). Event-based traceability for managing evolutionary change. *IEEE Trans. Softw. Eng.*, 29(9):796–810.

Cleland-Huang, J., Chang, C. K., and Christensen, M. (2003b). Event-based traceability for managing evolutionary change. *IEEE Tran on Soft Eng*, 29(9):796–810.

Cleland-Huang, J., Czauderna, A., Gibiec, M., and Emenecker, J. (2010). A machine learning approach for tracing regulatory codes to product specific requirements. In *ICSE Conference*, pages 155–164, New York, NY, USA. ACM.

Cleland-Huang, J., Settimi, R., BenKhadra, O., Berezhanskaya, E., and Christina, S. (2005). Goal-centric traceability for managing non-functional requirements. In *ICSE Conference*, pages 362–371, New York, NY, USA. ACM.

Cleland-Huang, J., Zemont, G., and Lukasik, W. (2004). A heterogeneous solution for improving the return on investment of requirements traceability. In *IEEE RE Conference*, pages 230–239, Washington, DC, USA. IEEE Computer Society.

Cooper, W. S. (1968). Expected search length: A single measure of retrieval effectiveness based on the weak ordering action of retrieval systems. *J. Am. Soc. Inf. Sci.*, 19(1):30–41.

De Lucia, A., Fasano, F., Oliveto, R., and Tortora, G. (2004). Enhancing an artifact management system with traceability recovery features. In *20th IEEE ICSM Conference*, pages 306–315, Washington, DC, USA. IEEE Computer Society.

De Lucia, A., Oliveto, R., and Tortora, G. (2008). Adams re-trace: Traceability link recovery via latent semantic indexing. In *Proceedings of the 30th ICSE Conference*, ICSE '08, pages 839–842, New York, NY, USA. ACM.

Deursen, A. V., Stehouwer, A., Lormans, M., Lormans, M., gerhard Gross, H., gerhard Gross, H., Solingen, R. V., and Solingen, R. V. (2006). Monitoring requirements coverage using reconstructed views: An industrial case study. In *13th Working Conf. on Reverse Eng*, pages 275–284.

Egyed, A. (2003). A scenario-driven approach to trace dependency analysis. *IEEE Trans. Softw. Eng.*, 29(2):116–132.

Finkelstein, A. (1994//). Requirements engineering: a review and research agenda. pages 10 – 19, Los Alamitos, CA, USA.

Gonzalez, R. C. and Woods, R. E. (2006). *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Hayes, J. H., Dekhtyar, A., and Osborne, J. (2003). Improving requirements tracing via information retrieval. In *20th IEEE RE Confenrece*, pages 151–161.

Hayes, J. H., Dekhtyar, A., and Sundaram, S. (2005). Text mining for software engineering: How analyst feedback impacts final results. *SIGSOFT Softw. Eng. Notes*, 30(4):1–5.

Hayes, J. H., Dekhtyar, A., Sundaram, S., Holbrook, A., Vadlamudi, S., and April, A. (2008). Requirements tracing on target (retro): Improving software maintenance through traceability recovery.

Hayes, J. H., Dekhtyar, A., and Sundaram, S. K. (2006a). Advancing candidate link generation for requirements tracing: The study of methods. *IEEE Tran on Soft Eng*, 32:4–19.

Hayes, J. H., Dekhtyar, A., and Sundaram, S. K. (2006b). Advancing candidate link generation for requirements tracing: the study of methods. *IEEE Tran on Soft Eng*, 32(1):4–19.

Jackson, P. and Moulinier, I. (2002). *Natural language processing for online applications. Text retrieval, extraction and categorization*, volume 5. Benjamins, Amsterdam, Philadelphia.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques.

Karlsson, L., Dahlstedt, s. G., Regnell, B., Natt och Dag, J., and Persson, A. (2007). Requirements engineering challenges in market-driven software development - an interview study with practitioners. *Inf. Softw. Technol.*, 49(6):588–604.

Kekäläinen, J. and Järvelin, K. (2002). Using graded relevance assessments in ir evaluation. *J. Am. Soc. Inf. Sci. Technol.*, 53(13):1120–1129.

Kochhar, P. S., Xia, X., Lo, D., and Li, S. (2016). Practitioners' expectations on automated fault localization. In *Proceedings of the 25th International Symposium on Software Testing and Analysis*, pages 165–176. ACM.

Konrad, S. and Gall, M. (2008). Requirements engineering in the development of large-scale systems. In *Proceedings of the 16th International Requirements Engineering Conference (RE 2008)*, pages 217–222.

Lee, M. D., Pincombe, B., and Welsh, M. (2007). A comparison of machine measures of text document similarity with human judgments.

Leveson, N. G. (1997). Software engineering: stretching the limits of complexity. *Commun. ACM*, 40(2):129–131.

Lin, J., Lin, C. C., Cleland-Huang, J., Settimi, R., Amaya, J., Bedford, G., Berenbach, B., Khadra, O. B., Duan, C., and Zou, X. (2006). Poirot: A distributed tool supporting enterprise-wide automated traceability. In *14th IEEE International Requirements Engineering Conference (RE'06)*, pages 363–364.

Maccari, A. (1999//). The challenges of requirements engineering in mobile telephones industry. pages 336 – 9, Los Alamitos, CA, USA.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Manning, C. D. and Schütze, H. (2002). *Foundations of Statistical Natural Language Processing*. MIT Press.

Marcus, A. and Maletic, J. (2003). Recovering documentation-to-source-code traceability links using latent semantic indexing. In *ICSE Conference*, pages 125–135, Washington, DC, USA. IEEE Computer Society.

Murta, L. G. P., van der Hoek, A., and Werner, C. M. L. (2006). Archtrace: Policy-based support for managing evolving architecture-to-implementation traceability links. In *ASE Conference*, pages 135–144, Washington, DC, USA. IEEE Computer Society.

Myaeng, S. H. and Korfhage, R. R. (1990). Integration of user profiles: models and experiments in information retrieval. *Information Processing & Management*, 26(6):719 – 738.

Natt och Dag, J. (2006a). *Managing Natural Language Requirements in Large-Scale Software Development*. PhD thesis, Lund University.

Natt och Dag, J. (2006b). The reqsimile tool website. `http://reqsimile.sourceforge.net/`.

Natt och Dag, J., Thelin, T., and Regnell, B. (2006). An experiment on linguistic tool support for consolidation of requirements from multiple sources in market-driven product development. *Empirical Software Engineering*, 11(2):303–329.

Northrop, L., Felier, P., Habriel, R. P., Boodenough, J., Linger, R., Klein, M., Schmidt, D., Sullivan, K., and Wallnau, K. (2006). *Ultra-Large-Scale Systems: The Software Challenge of the Future*. Software Engineering Institute.

och Dag, J. N., Gervasi, V., Brinkkemper, S., and Regnell, B. (2004). Speeding up requirements management in a product software company. *RE Conference*, 0:283–294.

och Dag, J. N., Gervasi, V., Brinkkemper, S., and Regnell, B. (2005). A linguistic-engineering approach to large-scale requirements management. *IEEE Softw.*, 22(1):32–39.

Pollack, S. M. (1968). Measures for the comparison of information retrieval systems. *Am. Doc.*, 19(4):387–397.

Raja, U. A. and Kamran, K. (2008). Framework for requirements traceability.

Raol, J. R. (2009). *Multi-Sensor Data Fusion with MATLAB: Theory and Practice*. Taylor and Francis, Inc.

Rath, M., Lo, D., and Mäder, P. (2018). Analyzing requirements and traceability information to improve bug localization. In *Int Conf Mining Software Repositories (MSR)*, pages 442–453. IEEE.

Rocchio, J. J. J. (1966). *Document retrieval systems: optimization and evaluation*. PhD thesis, Harvard University, USA.

Russ, J. C. (1999). *The image processing handbook (3rd ed.)*. CRC Press, Inc., Boca Raton, FL, USA.

Sabaliauskaite, G., Loconsole, A., Engström, E., Unterkalmsteiner, M., Regnell, B., Runeson, P., Gorschek, T., and Feldt, R. (2010). Challenges in aligning requirements engineering and verification in a large-scale industrial context. In *Proc. REFSQ 2010*.

Sabetzadeh, M. and Easterbrook, S. (2005). Traceability in viewpoint merging: A model management perspective.

Spanoudakis, G., Zisman, A., Perez-Minana, E., and Krause, P. (2004). Rule-based generation of requirements traceability relations. *JSS*, 72(2):105–127.

Spink, A. and Greisdorf, H. (2001). Regions and levels: measuring and mapping users' relevance judgments. *J. Am. Soc. Inf. Sci. Technol.*, 52(2):161–173.

Stathaki, T. (2008). *Image Fusion: Algorithms and Applications*. Academic Press.

Tariq Banday, M. and Jan, T. R. (2009). Effectiveness and Limitations of Statistical Spam Filters. *ArXiv e-prints*.

Vitányi, P. M. B., Balbach, F. J., Cilibrasi, R. L., and Li, M. (2008). *Information Theory and Statistical Learning*, chapter Normalized Information Distance, pages 45–82. Springer.

Yin, R. K. (2002). *Case Study Research: Design and Methods*. Sage Publications.

Zhou, J., Zhang, H., and Lo, D. (2012). Where should the bugs be fixed? more accurate information retrieval-based bug localization based on bug reports. In *34th ICSE Conference*, pages 14–24.