2IMP25 - Software Evolution

Requirement Traceability

Quartile 3 - 2019-2020

Group 50

Full Name	Student ID	Email
Yu Zhong	1416804	y.zhong@student.tue.nl
Xin Zhao	1329553	x.zhao1@student.tue.nl

Eindhoven, February 19, 2020

Contents

1 Introduction

After a system established, huge efforts have to be paid for its maintaining and evolution. As the environment changes, requirements need to be adapted and even a simple modification of requirement document may lead to a ripple effect on system implementation. Requirement traceability can benefit this co-evolution process by describing and following the life of a requirement in both forward and backward directions [3], through which engineers can understand where a requirement came from, how it was implemented and how it was tested. Thus, it is useful to determine what design, code, and test cases need to be updated to fulfill a change request for requirement documents made during the evolution phase [2].
Traceability can be realized by capturing trace-link manually or automatically. However, manually capturing traceability links for large software project means extreme amount of time and effort since the links that need to be captured grow exponentially [1]. Moreover, manual methods are error-prone and vulnerable to changes in the system [2]. Because of these shortcomings, manual methods are not suitable for any software systems other than a small one and hence automated tools are widely considered and used to detect trace-links. Clearly, traceability tools could significantly reduce the amount of time and effort that software engineers or organizations spend on trace-link detection. Meanwhile, the tools can provide quality traceability with less errors.
In this assignment, we focus on developing a traceability detection tool, tracing high-level requirements to low-level requirements, and evaluating the tool we developed on two group of Waterloo dataset. Through this assignment, we capture the concept about traceability and the process of trace-link detection. The report is structured as six sections. In Section 2, we briefly illustrate how we implemented the tool and which components we chose. In Section 4, by comparing the result given by our tool and the actual trace-links, we evaluate our tool and analysis the performance and then further discuss the reasons and limitations in Section 5.

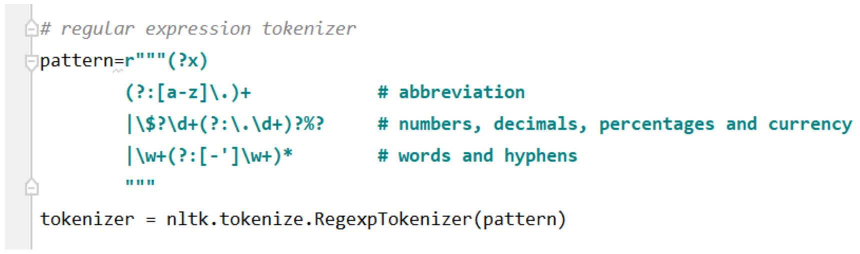
2 Trace-link detection tool

2.1 Pre-processing

The given requirement documents are stored in the form of *.csv* file and every requirement consists of two attributes, requirement id and content. For convenience, we used the library *Pandas* in implementation to realize IO operation, data manipulation and analysis. The function *read_csv*() from *Pandas* library is used to read files and store data as *Dataframe* type.
After getting the data from *.csv* file, several techniques are needed in the pre-processing phase, including parsing and tokenizing the content of requirements, removing stop-words in the token lists and extracting the stem words. All the functionalities above can be found in library *nltk*. The function implementing the pre-processing step is called *Preprocess*().

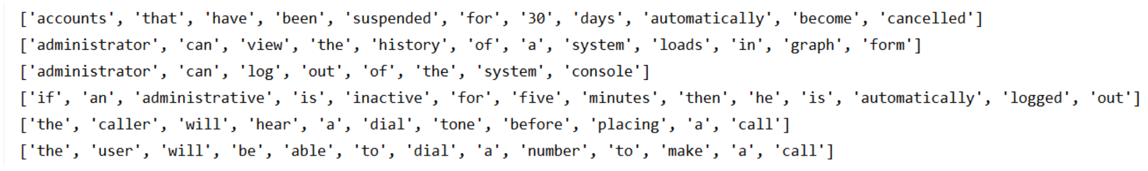
2.1.1 Parsing and Tokenizing

For the purpose of detecting the relations between high-level and low-level requirements, the very first step is to parse the content of requirements to a list of tokens. This is done with *RegexpTokenizer* from *nltk*. The *RegexpTokenizer* accepts a regular expression that matches tokens as a parameter. The regular expression and the use of tokenizer are shown in Figure 1 and the result of tokenizing can be seen in Figure 2. Note that before tokenizing the context of requirements, *.lower*() is called to transform capital letters into lowercases.



```
# regular expression tokenizer
pattern=r"""(?x)
        (?:[a-z]\.)+              # abbreviation
        |\$?\d+(?:\.\d+)?%?       # numbers, decimals, percentages and currency
        |\w+(?:[-']\w+)*          # words and hyphens
        """
tokenizer = nltk.tokenize.RegexpTokenizer(pattern)
```

Figure 1: Regular expression tokenizer

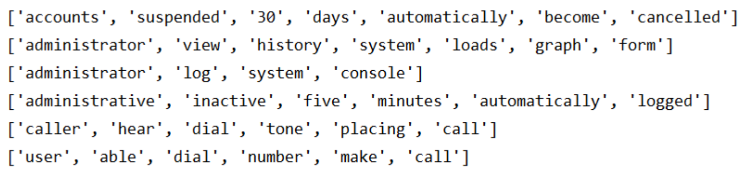


```
['accounts', 'that', 'have', 'been', 'suspended', 'for', '30', 'days', 'automatically', 'become', 'cancelled']
['administrator', 'can', 'view', 'the', 'history', 'of', 'a', 'system', 'loads', 'in', 'graph', 'form']
['administrator', 'can', 'log', 'out', 'of', 'the', 'system', 'console']
['if', 'an', 'administrative', 'is', 'inactive', 'for', 'five', 'minutes', 'then', 'he', 'is', 'automatically', 'logged', 'out']
['the', 'caller', 'will', 'hear', 'a', 'dial', 'tone', 'before', 'placing', 'a', 'call']
['the', 'user', 'will', 'be', 'able', 'to', 'dial', 'a', 'number', 'to', 'make', 'a', 'call']
```

Figure 2: The result of tokenizing for some requirements

2.1.2 Stop-words Removal

After parsing and tokenizing we had to remove all the words that we had no interest in. Stop-words including articles and auxiliary verbs do not really express relation between requirements, and hence should be eliminated. To this end, we took advantage of *stopwords* module from package *nltk.corpus*, and utilized the stop-words glossary directly. For each token in the token list, we removed it if it is in the stop-words glossary. Figure 3 shows the results of stop-words removal for token lists in Figure 2

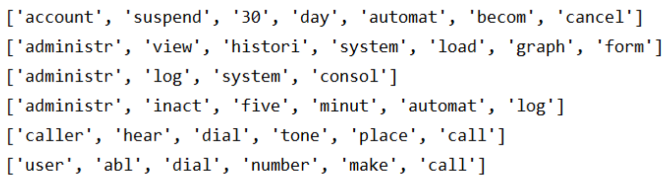


```
['accounts', 'suspended', '30', 'days', 'automatically', 'become', 'cancelled']
['administrator', 'view', 'history', 'system', 'loads', 'graph', 'form']
['administrator', 'log', 'system', 'console']
['administrative', 'inactive', 'five', 'minutes', 'automatically', 'logged']
['caller', 'hear', 'dial', 'tone', 'placing', 'call']
['user', 'able', 'dial', 'number', 'make', 'call']
```

Figure 3: The result of stop-words removal for token lists in Figure 2

2.1.3 Stemming

Continuing on, we extracted stemmed words from the token list. Since nltk encapsulates the functionality of Snowball library, we initialized an English $Poster2$ stemmer by function $EnglishStemmer()$ in module *nltk.stem.snowball* directly. Poster2 stemming algorithm is less complex than the algorithm developed by Lovins and it addresses the ”overstem” problem in Poster stemming algorithm, which make it reasonable to utilize Poster2 stemmer in our implementation. Figure 4 shows the result of stemming for token lists in Figure 3.

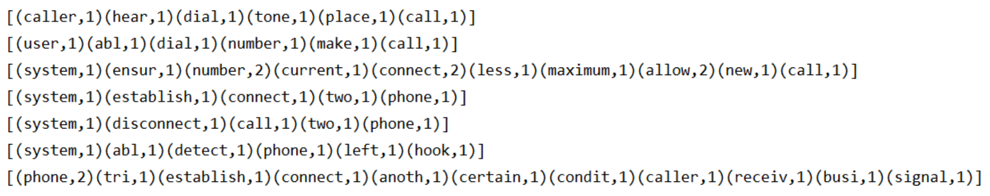


```
['account', 'suspend', '30', 'day', 'automat', 'becom', 'cancel']
['administr', 'view', 'histori', 'system', 'load', 'graph', 'form']
['administr', 'log', 'system', 'consol']
['administr', 'inact', 'five', 'minut', 'automat', 'log']
['caller', 'hear', 'dial', 'tone', 'place', 'call']
['user', 'abl', 'dial', 'number', 'make', 'call']
```

Figure 4: The result of stop-words removal for token lists in Figure 3

2.1.4 Frequency Distribution

A frequency distribution records the number of times each token has occurred in the token list, which is the preparation for computing vector representation in Section 2.2. The *nltk* library provides a function called $FreqDist()$ which accepts a list of token for a requirement and returns a dictionary with keys and values referring to token and frequency of this token respectively. The result of frequency distribution calculation can be seen as Figure 5.



```
[(caller,1)(hear,1)(dial,1)(tone,1)(place,1)(call,1)]
[(user,1)(abl,1)(dial,1)(number,1)(make,1)(call,1)]
[(system,1)(ensur,1)(number,2)(current,1)(connect,2)(less,1)(maximum,1)(allow,2)(new,1)(call,1)]
[(system,1)(establish,1)(connect,1)(two,1)(phone,1)]
[(system,1)(disconnect,1)(call,1)(two,1)(phone,1)]
[(system,1)(abl,1)(detect,1)(phone,1)(left,1)(hook,1)]
[(phone,2)(tri,1)(establish,1)(connect,1)(anoth,1)(certain,1)(condit,1)(caller,1)(receiv,1)(busi,1)(signal,1)]
```

Figure 5: The result of frequency distribution calculation for some requirements

By now, all the pre-processing steps are performed. The data structure we used to store the outcomes of pre-processing is a dictionary with its keys storing the id of requirements and its values storing the frequency distribution of tokens in the corresponding requirement.

2.2 Vector Representation

For the purpose of matching key words inside requirements and measure the similarity of key words, vector representation in uniform format is necessary for all requirements. To achieve this goal, we need to make use of the master vocabulary, a set of all tokens (that after stemming) appearing in all requirements. Given a master vocabulary $\langle W_1, W_2, ..., W_N \rangle$, the vector representation v_r of a requirement r is calculated as follow.

$$v_r = \langle w_1, w_2, ..., w_N \rangle \quad \text{where } w_i = 0 \text{ if } W_i \text{ does not appear in the requirement } r$$
$$\text{or } w_i = tf \times idf \text{ if } W_i \text{ appears in the requirement } r$$

The tf and idf refer to term frequency and inverse document frequency respectively.

$$tf = \text{The number of times } W_i \text{ has occurred in } r$$
$$idf = log_2(\frac{n}{d}) \text{ where } n \text{ is the total number of requirements}$$
$$\text{and } d \text{ is the number of requirements containing } W_i$$

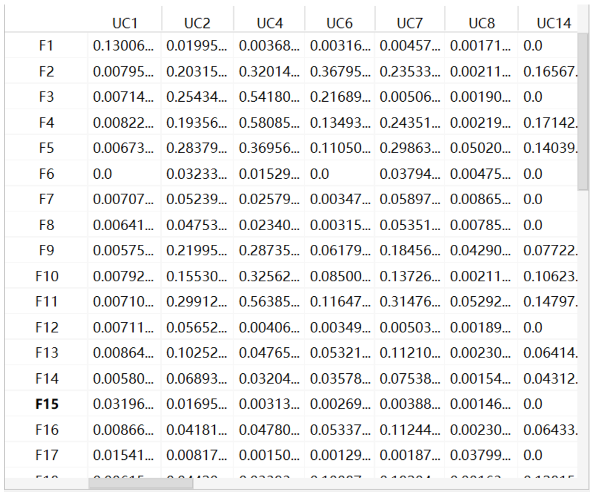
Instead of constructing a master vocabulary, we constructed dictionaries for term frequency and inverse document frequency taking terms in master vocabulary as keys. The tf value is the same as frequency distribution value as we calculated in Section 2.1.4. The idf value is calculated in function called $vector\_representation()$.

2.3 Similarity Matrix

Given M high-level $(h_1, h_2, ..., h_M)$ and P low-level $(l_1, l_2, ..., l_P)$ vector representations for requirements with N words in the master vocabulary, the similarity matrix is a $M \times P$ Matrix where the value in each cell is calculated by the formula as follow.

$$\begin{aligned} sim(h,l) &= cos(h,l) \\ &= cos(\langle x_1, x_2, ..., x_N \rangle, \langle y_1, y_2, ..., y_N \rangle) \\ &= \frac{\sum_{i=1}^{N} x_i \times y_i}{\sqrt{\sum_{i=1}^{N} {x_i}^2}\sqrt{\sum_{i=1}^{N} {y_i}^2}} \end{aligned}$$

The similarity value for a pair of requirements is calculated by function *get_similarity*(). Figure 6 shows the similarity matrix we got.



	UC1	UC2	UC4	UC6	UC7	UC8	UC14
F1	0.13006...	0.01995...	0.00368...	0.00316...	0.00457...	0.00171...	0.0
F2	0.00795...	0.20315...	0.32014...	0.36795...	0.23533...	0.00211...	0.16567.
F3	0.00714...	0.25434...	0.54180...	0.21689...	0.00506...	0.00190...	0.0
F4	0.00822...	0.19356...	0.58085...	0.13493...	0.24351...	0.00219...	0.17142.
F5	0.00673...	0.28379...	0.36956...	0.11050...	0.29863...	0.05020...	0.14039.
F6	0.0	0.03233...	0.01529...	0.0	0.03794...	0.00475...	0.0
F7	0.00707...	0.05239...	0.02579...	0.00347...	0.05897...	0.00865...	0.0
F8	0.00641...	0.04753...	0.02340...	0.00315...	0.05351...	0.00785...	0.0
F9	0.00575...	0.21995...	0.28735...	0.06179...	0.18456...	0.04290...	0.07722.
F10	0.00792...	0.15530...	0.32562...	0.08500...	0.13726...	0.00211...	0.10623.
F11	0.00710...	0.29912...	0.56385...	0.11647...	0.31476...	0.05292...	0.14797.
F12	0.00711...	0.05652...	0.00406...	0.00349...	0.00503...	0.00189...	0.0
F13	0.00864...	0.10252...	0.04765...	0.05321...	0.11210...	0.00230...	0.06414.
F14	0.00580...	0.06893...	0.03204...	0.03578...	0.07538...	0.00154...	0.04312.
F15	0.03196...	0.01695...	0.00313...	0.00269...	0.00388...	0.00146...	0.0
F16	0.00866...	0.04181...	0.04780...	0.05337...	0.11244...	0.00230...	0.06433.
F17	0.01541...	0.00817...	0.00150...	0.00129...	0.00187...	0.03799...	0.0

Figure 6: Similarity Matrix

2.4 Trace-link Generation

Based on the similarity matrix, we used the following baseline techniques for further filtering of the trace-link, and the fourth technique is the custom one:

1. >0 filtering: for every h, report all l with the similarity value of at least 0.
2. At least 0.25: for every h, report all l with the similarity value of at least 0.25.
3. At least $0.67 \times max_{l \in L}(sim(h,l))$: for every h, report all l with the similarity value of at least $0.67 \times max_{l \in L}(sim(h,l))$.
4. At least $0.67 \times max_{l \in L}(sim(h,l))$ with UB=2: for every h, report the largest one or two l (take 2 as the upper bound) with the similarity value of at least $0.67 \times max_{l \in L}(sim(h,l))$.

Function *link_generation*() accepts an integer referring to mode selection (1: no filtering, 2: at least 0.25, 3: at least 0.67max, 4: at least 0.67max with the upper bound 2) as a parameter, and generates a *DataFrame* containing high-level requirements and their linked low-level requirements. We used the function *to_csv*() from *Pandas* to export the links generated by different filters, and store them in separate files. We discuss the implementation of quality evaluation of these results in Section 4.

2.5 Run

In order to run the tool, the following commands should be used:

- docker build -t 2imp25-assignment-1 ./
- docker run –rm -v "$pwd\dataset-1:/input" -v "$pwd\output:/output" 2imp25-assignment-1 0

The final integer in the command is the baseline technique selection while 0 refers to implementing all four baseline techniques and 1, 2, 3, 4 refer to implementing techniques shows in 2.4 respectively.

3 Dataset

The dataset we used is the Waterloo dataset that consists of projects completed by students in University of Waterloo. The students were given the task of designing a voice-over-IP management software system [4]. For the purpose of evaluating our trace-link detection tool, we made use of two documents in two (Project 1 and Project 9) of these variants: requirement specification and the use case description. In each project, high-level requirements are stated in *high.csv* while use cases are stated in *low.csv*. The file *links.csv* contains manually recognized trace-links between the functional requirements and use case description.
An example functional requirement in *high.csv* of Project 1 follows: *F35, The caller will hear a dial tone before placing a call.* $F35$ is the id of the requirement. According to *links.csv*, an example use case that link this requirement is: *UC36, This use case captures the process by which the caller places a call to the callee. The caller picks up the phone, receives a dial tone, and then proceeds to dial 4 digits to make the call* (see *low.csv* for detail description).

4 Evaluation

Since the tool might fail to detect some trace-links or identify some spurious trace-links, we took advantage of the confusion matrix and the F-measure value to evaluate the performances of baseline techniques for generating trace-links as stated in Section 2.4. The confusion matrix is a table that describe the performance of a model on a set of test data for which the true value is known. The matrix consists of four types of value: true positive, true negative, false positive, false negative; and the F-measure values is calculated based on these four values.

	NTL predicted by the tool	NTL not predicted by the tool
NTL identified manually	$V_{true\ positive}$	$V_{false\ negative}$
NTL not identified manually	$V_{false\ positive}$	$V_{true\ negative}$

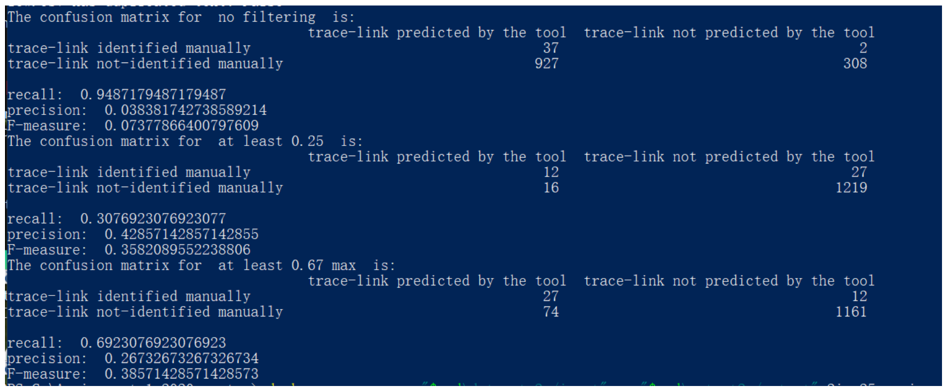
Table 1: The sample of the confusion matrix (NTL refers to the number of trace-links)

In this assignment, the confusion matrix is shown as Table 1 (note that NTL in the table refers to the number of trace-link) and the F-measure value is calculated as follows.

$$\begin{aligned} Recall\ value: \ r &= \frac{V_{true\ positive}}{V_{true\ positive} + V_{true\ negative}} \\ Precision\ value: \ p &= \frac{V_{true\ positive}}{V_{true\ positive} + V_{false\ positive}} \\ F\text{-}measure\ value: \ f &= 2 \times \frac{p \times r}{p + r} \end{aligned}$$

The evaluation process is implemented in the function called *evaluation*(), which can print out the confusion matrix and F-measure values for a given trace-link result predicted by our tool. Figure

7 shows an example of the output of evaluation for *dataset*-1. In this section we will discuss the performance in detail.



```
The confusion matrix for  no filtering  is:
                                        trace-link predicted by the tool  trace-link not predicted by the tool
trace-link identified manually                                        37                                     2
trace-link not-identified manually                                   927                                   308

recall:  0.9487179487179487
precision:  0.038381742738589214
F-measure:  0.07377866400797609
The confusion matrix for  at least 0.25  is:
                                        trace-link predicted by the tool  trace-link not predicted by the tool
trace-link identified manually                                        12                                    27
trace-link not-identified manually                                    16                                  1219

recall:  0.3076923076923077
precision:  0.42857142857142855
F-measure:  0.3582089552238806
The confusion matrix for  at least 0.67 max  is:
                                        trace-link predicted by the tool  trace-link not predicted by the tool
trace-link identified manually                                        27                                    12
trace-link not-identified manually                                    74                                  1161

recall:  0.6923076923076923
precision:  0.26732673267326734
F-measure:  0.38571428571428573
```

Figure 7: The evaluation output for *dataset*-1

4.1 The confusion matrix for >0 filtering

Using the baseline trace-link generation technique that reports the links with similarity value of at least >0, we get the following evaluation results for *dataset*-1 (see Table 2) and *dataset*-2 (see Table 3).

	NTL predicted by the tool	NTL not predicted by the tool
NTL identified manually	37	2
NTL not identified manually	927	308

Table 2: The confusion matrix for *dataset*-1 with no filtering baseline technique

	NTL predicted by the tool	NTL not predicted by the tool
NTL identified manually	57	31
NTL not identified manually	821	851

Table 3: The confusion matrix for *dataset*-2 with no filtering baseline technique

As we can observe, this baseline technique will fail to report some actual trace-links or report some spurious trace-links. The undetected trace-links (false negative) and spuriously detected trace-links (false positive) can be found in Appendices A.1 for *dataset*-1 and in Appendices B.1 for *dataset*-2. As we stated in Section 2.4, the trace-links between a pair of requirements will not be generated by this baseline technique if and only if their similarity score = 0, which means that there is no duplicated token in their vector representation. Hence, for example, if a pair of requirements avoid containing same tokens, by using synonyms, the tool will not report the link between them even if it actually exists. A clear example of this is the undetected trace-links for *dataset*-1. The trace-links $F6-UC22$, $F6-UC23$ are not reported because they do not share any same token.

- F6: Grant accounts the ability to originate and receive calls.
- UC22: Administrator wishes to turn on a customer's filter. Customer ID, filter to turn on as input. The administrator turns on either the black list filter or the white list filter. The filter

selection is mutually exclusive; either the black filter is on, the white filter is on, or no filter is on.
- UC23: Administrator wishes to turn off all filters on customer's phone. Customer ID as input. The administrator turns off the filter on the customer's phone.

On the contrary, the link between any pair of requirements with any shared token will be generated by this baseline technique, which has consequence to a large amount of spuriously detected trace-links.

4.2 The confusion matrix for at least 0.25

Using the baseline trace-link generation technique that reports the links with similarity value of at least 0.25, we get the following evaluation results for *dataset*-1 (see Table 4) and *dataset*-2 (see Table 5). The undetected trace-links (false negative) and spuriously detected trace-links (false positive) can be found in Appendices A.2 for *dataset*-1 and in Appendices B.2 for *dataset*-2.

	NTL predicted by the tool	NTL not predicted by the tool
NTL identified manually	20	19
NTL not identified manually	38	1197

Table 4: The confusion matrix for *dataset*-1 with at least 0.25 baseline technique

	NTL predicted by the tool	NTL not predicted by the tool
NTL identified manually	18	70
NTL not identified manually	23	1649

Table 5: The confusion matrix for *dataset*-2 with at least 0.25 baseline technique

We can observe that by using this baseline technique, the number of spuriously detected trace-links reduces significantly, comparing to the result shown in Section 4.1. Moreover, this technique fails to detect actual trace-links more often. It is trivial to reason about this since the threshold increases from >0 to ≥ 0.25 and thus the trace-links will not be detected if the high-level and low-level requirements only share limit amount of tokens or share some commonly used token. The undetected trace-link $F4 - UC7$ in *dataset*-1 is an example of this.

- F4: Administrator can associate an IP address to a phone.
- UC7: Administrator wishes to assign a phone number to a customer's phone, phone number, account ID as input. The administrator assigns a phone number to a customer's account.

It is clear that these two requirements only share tokens *administr* and *phone*, and they are commonly used in requirement documents. We will discuss in details in Section 5.

4.3 The confusion matrix for at least 0.67max

Using the baseline trace-link generation technique that reports the links with similarity value of at least 0.67max, we get the following evaluation results for *dataset*-1 (see Table 6) and *dataset*-2 (see Table 7).The undetect trace-links (false negative) and spuriously detected trace-links (false positive) can be found in Appendices A.3 for *dataset*-1 and in Appendices B.3 for *dataset*-2.

	NTL predicted by the tool	NTL not predicted by the tool
NTL identified manually	23	16
NTL not identified manually	61	1174

Table 6: The confusion matrix for *dataset*-1 with at least 0.67max baseline technique

	NTL predicted by the tool	NTL not predicted by the tool
NTL identified manually	37	51
NTL not identified manually	125	1547

Table 7: The confusion matrix for *dataset*-2 with at least 0.67max baseline technique

This baseline technique provides every high-level requirement with a flexible threshold which contributes to less undetected trace-links compared with the baseline technique $\geq$0.25. However, this technique also guarantees at least one trace-link for every high-level requirement and hence leads to more spuriously identified trace-links.

4.4 The confusion matrix for at least 0.67max with UB=2

Using the baseline trace-link generation technique that reports the largest two links with similarity value of at least 0.67max for every high-level requirements, we get the following evaluation results for *dataset*-1 (see Table 8) and *dataset*-2 (see Table 9). The undetect trace-links (false negative) and spuriously detected trace-links (false positive) can be found in Appendices B.4 for *dataset*-1 and in Appendices **??** for *dataset*-2.

	NTL predicted by the tool	NTL not predicted by the tool
NTL identified manually	22	17
NTL not identified manually	48	1187

Table 8: The confusion matrix for *dataset*-1 with at least 0.67max and UB=2 baseline technique

	NTL predicted by the tool	NTL not predicted by the tool
NTL identified manually	36	52
NTL not identified manually	91	1581

Table 9: The confusion matrix for *dataset*-2 with at least 0.67max and UB=2 baseline technique

We observed that in the previous section the result given by $\geq$0.67max is not precise enough due to the large amount of spurious trace-links. Motivated by this problem, we came out with a upper bound which restricts the number of links for each high-level requirement. As the tables show, the way that add an upper bound to the number of trace-links generated for every high-level requirements slightly improves the performance of baseline technique $\geq$0.67max by reducing the number of spuriously detected trace-links.
However, how to decide the upper bound is tricky since prior knowledge about trace-links distribution is needed. In our cases, the number of trace-links for most high-level requirements are roughly bounded

by 2.
It is easy to observe that some high-level requirements have more than 2 trace-links and except for the largest two, the rest would be ignored by the tool, i.e., the undetected links $F5 - UC2$ in *dataset*-1 because of the detected trace-links $F5 - UC4, UC7$. On the other hand, some high-level requirements may have no corresponding trace-links or only one trace-link. Similarly to ≥0.67max filtering, this leads to many spuriously detected trace-links .

4.5 Comparison

Table 10 and 11 compare different baseline techniques in terms of the recall, precision and F-measure values for *dataset*-1 and *dataset*-2.

	>0 filtering	≥0.25	≥0.67max	≥0.67max with UB=2
Recall	0.948718	0.512821	0.589743	0.564103
Precision	0.038382	0.344828	0.273810	0.314286
F-measure	0.073779	0.412371	0.373984	0.403670

Table 10: The recall, precision and F-measure values for *dataset*-1 with different baseline techniques

	> 0 filtering	≥0.25	≥0.67max	≥0.67max with UB=2
Recall	0.647727	0.204545	0.420455	0.409091
Precision	0.064920	0.439024	0.228395	0.283465
F-measure	0.118012	0.279069	0.296	0.334884

Table 11: The recall, precision and F-measure values for *dataset*-2 with different baseline techniques

The tables show that the baseline technique with > 00 filtering gets the highest recall score among these four techniques but has the lowest precision score and f-measure value. The techniques with ≥0.25 and ≥0.67max have a similar level of performance in terms of F-measure value, and the former performs better in precision while the latter gets a better score in recall. Our technique is the improved version of the technique ≥0.67max. It improves the precision score while trying to keep the same recall value.
The misclassifications made by these techniques share some common grounds:

1. These techniques cannot identify the trace-link between two requirements if they do not have any same token. As we stated in Section 4.1, the trace-links $F6 - UC22, UC23$ cannot be detected by any of these techniques.
2. On the contrary, these techniques will spuriously identify a trace-link if two requirements share some tokens. For example the spuriously detected trace-link $F4 - UC4$ in *dataset*-1 is generated based on shared tokens: *administr*, *IP*, *address*, *associat*, *phone*.
   - F4: Administrator can associate an IP address to a phone.
   - UC4: Administrator wishes to find the IP address of a new customer's phone. The administrator finds the IP address of a new customer's phone in order to associated it with the account.

Apart from the similarities, there are certainly some differences in the misclassifications of these techniques.

1. Compared with the baseline techniques with >0 filtering, other techniques have stricter thresholds and will fail to detect the relation between requirements that only share some commonly used tokens, such as *administr*, *phone*, *account* in this case, as we stated in Section 4.2.

2. The baseline technique $\geq$0.67max guarantees at least one trace-link per high-level requirement and detects more spurious trace-links in our case.
3. Our solution guarantees that a high-level requirement will not have no more than 2 trace-links. Hence, it will result in less spurious trace-links compared to the technique $\geq$0.67max.

5 Discussion

From Section 4.1 to 4.5 we noticed that the trace-link detection tool is flawed. A main problem is that it cannot detect trace-links between requirements that do not have same tokens or only share some commonly used tokens which have relatively low *idf* values and finally result at lower similarity score. A way we suggest to address this problem is to take an extra glossary of synonyms or the logically linking words in this domain and take it into account when calculating the similarity score.
Another problem is about spuriously detected trace-links. This usually happens when two requirements shares a certain number of tokens and this will even be misclassified by human at the first glance, not to mention programs. We did not find a effective way to actually tackle this problem but we can set an upper bound for the number of links per high-level requirement together with the filter to reduce the number of spuriously detected trace-links as what we did in the fourth trace-link generation technique.
In addition, we noticed that these techniques are suitable for different dataset for the purpose of achieving better performance. For example, the technique $\geq$0.67max performs better on *dataset*-2 since almost every high-level requirement links to one or more low-level requirements, which fits the feature of this baseline technique. On the other hand, the technique with $\geq$0.25 is more suitable for *dataset*-1 since over 30% of high-level requirements have no link to low-level requirements. However, this strategy can be tricky in practise because we know nothing about how the actual trace-links distribute.

6 Conclusion

In conclusion, we implemented a trace-link detection tool by pre-processing the data, creating the similarity matrix and implementing filtering techniques to map high-level requirements to low-level requirements and finally to compare and evaluate the results made by them.
During the evaluation, we noticed these techniques performance differently on dataset with different features and realised how flawed this tool can be no matter how we adjust our filter. A main problem is the failure to detected trace-links for requirements sharing the same meanings but using different words. A possible way to solve it is to provide an extra input, such as a glossary for synonyms. Another attempt is to decrease the number of spurious trace-links generated by our tool which is a tricky task. We made use of an upper bound to restrict the number of links reported but its performance still largely depends on the distribution of the actual trace-links.

References

[1] Jane Cleland-Huang, Carl Chang, and Christensen MJ. Event-based traceability for managing evolutionary change. *Software Engineering, IEEE Transactions on*, 29:796– 810, 10 2003.

[2] A Kannenberg and Hossein Saiedian. Why software requirements traceability remains a challenge. *The Journal of Defense Software Engineering*, 22:14–19, 07 2009.

[3] Balasubramaniam Ramesh, Curtis Stubbs, Timothy Powers, and Michael Edwards. Requirements traceability: Theory and practice. *Annals of Software Engineering*, 3(1):397–415, Jan 1997.

[4] Senthil Karthikeyan Sundaram, Jane Huffman Hayes, Alex Dekhtyar, and E Ashlee Holbrook. Assessing traceability of software engineering artifacts. *Requirements engineering*, 15(3):313–335, 2010.

Appendices

A Misclassification for Dataset-1

A.1 >0 filtering

Table 12 shows the list of undetected trace-links. Since the number of spuriously detected trace-links

Requirement id A	*Links* V
F6	”UC23 UC22”

Table 12: List for undetected trace-links for *dataset*-1 with >0 filtering

is too large and we cannot present these links by table within a page, we format them as follows:

- F46: UC29, UC1, UC2, UC7, UC27, UC42, UC22, UC15, UC4, UC18, UC17, UC19, UC35, UC24, UC36, UC21, UC8, UC23, UC16, UC6, UC25
- F9: UC18, UC1, UC22, UC16, UC2, UC8, UC21, UC14, UC23, UC6, UC29, UC7, UC24, UC17, UC36, UC4, UC35, UC19, UC27, UC25, UC42
- F7: UC15, UC16, UC6, UC36, UC25, UC8, UC42, UC4, UC27, UC2, UC7, UC29, UC18, UC17, UC1, UC19, UC35, UC40, UC24
- F23: UC15, UC25, UC8, UC2, UC35, UC24, UC4, UC22, UC27, UC18, UC1, UC36, UC23, UC21, UC17, UC7, UC16
- F47: UC2, UC18, UC7, UC8
- F12: UC18, UC16, UC6, UC19, UC4, UC27, UC25, UC2, UC22, UC24, UC15, UC35, UC42, UC7, UC1, UC8, UC36, UC17, UC29, UC23, UC21
- F5: UC4, UC14, UC6, UC25, UC1, UC31, UC17, UC19, UC33, UC22, UC29, UC24, UC15, UC36, UC35, UC42, UC7, UC23, UC16, UC27, UC21, UC32, UC40, UC18, UC8
- F31: UC19, UC2, UC4, UC36, UC16, UC17, UC25, UC8, UC18, UC7, UC15
- F14: UC25, UC36, UC29, UC4, UC35, UC16, UC23, UC21, UC33, UC14, UC17, UC7, UC15, UC40, UC42, UC2, UC6, UC22, UC24, UC19, UC18, UC1, UC8, UC27, UC31
- F37: UC6, UC19, UC7, UC1, UC8, UC18, UC21, UC2, UC4, UC15, UC14, UC42, UC24, UC25, UC17, UC36, UC16, UC35, UC40
- F38: UC14, UC23, UC1, UC33, UC15, UC4, UC6, UC21, UC18, UC16, UC35, UC2, UC31, UC40, UC32, UC25, UC24, UC36, UC7
- F43: UC4, UC14, UC18, UC42, UC15, UC23, UC24, UC32, UC7, UC16, UC31, UC40, UC21, UC36, UC2, UC1, UC6, UC33, UC17
- F3: UC22, UC7, UC8, UC21, UC29, UC19, UC23, UC18, UC35, UC2, UC16, UC1, UC17, UC42, UC6, UC36, UC25, UC15, UC24, UC27
- F20: UC22, UC36, UC4, UC25, UC42, UC1, UC17, UC19, UC24, UC29, UC16, UC18, UC35, UC8, UC15, UC6, UC7, UC21, UC2, UC23
- F6: UC4, UC40, UC8, UC17, UC15, UC7, UC25, UC36, UC19, UC2, UC16, UC42, UC18
- F36: UC24, UC42, UC18, UC6, UC21, UC15, UC40, UC14, UC16, UC2, UC7, UC17
- F39: UC40, UC15, UC24, UC17, UC42, UC6, UC31, UC16, UC21, UC7, UC18, UC1, UC2, UC33, UC36, UC32, UC14, UC35, UC25, UC23, UC4
- F18: UC32, UC15, UC2, UC33, UC25, UC8, UC6, UC42, UC21, UC17, UC23, UC18, UC16, UC36, UC35, UC7, UC14, UC4, UC40, UC29, UC1, UC22, UC31, UC19, UC27
- F42: UC7, UC2, UC16, UC8, UC4, UC17, UC25, UC27, UC21, UC22, UC23, UC15, UC18
- F16: UC24, UC36, UC1, UC27, UC23, UC2, UC6, UC15, UC8, UC7, UC32, UC17, UC42, UC16, UC4, UC25, UC35, UC22, UC19, UC14, UC18, UC40, UC21, UC33, UC29
- F48: UC35, UC29, UC27, UC21, UC15, UC25, UC18, UC36, UC19, UC7, UC1, UC2, UC6, UC22, UC23, UC42, UC16, UC24, UC4, UC8

- F40: UC6, UC1, UC25, UC4, UC35, UC15, UC40, UC14, UC33, UC7, UC2, UC36, UC31, UC16, UC23, UC24, UC32
- F17: UC6, UC18, UC8, UC21, UC27, UC42, UC29, UC22, UC36, UC24, UC4, UC2, UC23, UC19, UC25, UC16, UC1, UC35, UC15, UC17, UC7
- F26: UC29, UC18, UC6, UC42, UC24, UC35, UC4, UC27, UC1, UC8, UC22, UC2, UC23, UC7, UC21, UC15, UC19, UC17, UC16, UC36
- F44: UC6, UC31, UC42, UC32, UC2, UC23, UC16, UC4, UC15, UC14, UC7, UC18, UC40, UC33, UC21, UC17
- F32: UC23, UC18, UC19, UC6, UC36, UC1, UC4, UC42, UC29, UC21, UC25, UC35, UC16, UC2, UC22, UC27, UC7, UC8, UC17, UC15
- F11: UC21, UC18, UC36, UC22, UC27, UC1, UC32, UC6, UC14, UC16, UC19, UC42, UC2, UC23, UC29, UC25, UC35, UC31, UC40, UC17, UC4, UC15, UC7, UC8, UC33, UC24
- F49: UC27, UC8, UC4, UC36, UC17, UC7, UC35, UC1, UC18, UC21, UC29, UC2, UC16, UC40, UC15, UC6, UC24, UC19, UC22, UC25, UC23
- F30: UC4, UC36, UC1, UC7, UC29, UC2, UC24, UC17, UC42, UC6, UC18, UC21, UC22, UC23, UC25, UC27, UC16, UC8, UC35, UC19
- F41: UC24, UC16, UC40, UC2, UC32, UC35, UC17, UC15, UC4, UC14, UC19, UC31, UC33, UC6, UC7, UC23
- F45: UC16, UC40, UC15, UC19, UC35, UC21, UC27, UC29, UC17, UC8, UC18, UC23, UC25, UC36, UC4, UC22, UC24, UC6, UC2, UC42, UC7
- F1: UC27, UC4, UC21, UC19, UC18, UC35, UC29, UC6, UC7, UC8, UC15, UC22, UC36, UC2, UC17, UC25, UC24, UC16, UC23, UC42
- F2: UC1, UC36, UC29, UC32, UC33, UC15, UC22, UC4, UC23, UC24, UC25, UC27, UC21, UC19, UC2, UC42, UC35, UC17, UC14, UC18, UC8, UC40, UC7, UC16, UC31
- F29: UC18, UC6, UC4, UC21, UC24, UC35, UC42, UC7, UC27, UC19, UC8, UC2, UC1, UC15, UC25, UC36, UC22, UC17, UC29, UC23
- F27: UC17, UC25, UC21, UC16, UC35, UC18, UC24, UC2, UC29, UC6, UC36, UC27, UC8, UC4, UC7, UC1, UC42, UC22, UC23, UC15
- F22: UC16, UC22, UC4, UC24, UC23, UC36, UC15, UC25, UC7, UC27, UC17, UC2, UC21, UC18, UC8
- F35: UC17, UC40, UC18, UC16, UC21, UC42
- F21: UC2, UC27, UC1, UC36, UC8, UC25, UC24, UC18, UC35
- F34: UC25, UC2, UC22, UC19, UC18, UC29, UC1, UC6, UC16, UC35, UC23, UC17, UC15, UC4, UC42, UC24, UC21, UC8, UC7, UC36, UC27
- F15: UC35, UC25, UC24, UC19, UC36, UC22, UC27, UC42, UC21, UC4, UC16, UC1, UC18, UC8, UC6, UC17, UC15, UC23, UC2, UC7
- F8: UC36, UC25, UC1, UC40, UC24, UC2, UC8, UC16, UC19, UC27, UC29, UC15, UC7, UC18, UC35, UC17, UC42, UC4, UC6
- F28: UC4, UC16, UC19, UC1, UC25, UC21, UC24, UC17, UC22, UC7, UC35, UC27, UC42, UC18, UC8, UC36, UC29, UC15, UC6, UC23
- F33: UC42, UC2, UC19, UC18, UC8, UC22, UC25, UC16, UC7, UC17, UC27, UC29, UC23, UC1, UC15, UC36, UC24, UC21, UC4, UC6
- F24: UC36, UC24, UC2, UC19, UC23, UC35, UC7, UC17, UC16, UC4, UC21, UC18, UC22, UC25, UC42, UC15, UC29, UC6, UC1, UC27
- F10: UC42, UC22, UC25, UC24, UC2, UC35, UC4, UC6, UC8, UC15, UC17, UC36, UC29, UC16, UC19, UC18, UC27, UC21, UC23, UC7, UC1
- F13: UC24, UC4, UC15, UC14, UC32, UC31, UC42, UC17, UC2, UC23, UC18, UC8, UC7, UC29, UC40, UC35, UC6, UC27, UC22, UC21, UC16, UC25, UC36, UC1, UC19
- F4: UC35, UC1, UC29, UC32, UC24, UC18, UC17, UC21, UC36, UC8, UC22, UC4, UC31, UC27, UC33, UC14, UC19, UC15, UC23, UC42, UC16, UC2, UC6, UC25, UC40
- F19: UC1, UC36, UC35, UC27, UC18, UC24, UC42, UC21, UC2, UC16, UC25, UC17, UC40

- F25: UC1, UC24, UC42, UC22, UC2, UC25, UC15, UC27, UC21, UC35, UC17, UC8, UC6, UC36, UC23, UC19, UC29, UC4, UC7, UC16

A.2 At least 0.25

Table 13 shows the list of undetected trace-links and spurious trace-links.

Requirement id	*Links*
A	V
F32	UC24
F9	UC15
F1	UC1
F13	UC33
F6	"UC21 UC23 UC22"
F14	UC32
F7	"UC22 UC21 UC23"
F16	UC31
F49	UC42
F36	UC36
F4	UC7
F10	UC14
F8	"UC23 UC22 UC21"

(a) List for undetected trace-links

Requirement id	*Links*
A	V
F28	"UC8 UC25"
F11	"UC4 UC7 UC25 UC2"
F25	"UC2 UC19 UC8"
F5	"UC7 UC4"
F48	UC16
F24	"UC18 UC7 UC2 UC27"
F30	"UC16 UC25 UC2"
F33	UC1
F42	"UC27 UC8"
F3	UC2
F39	UC42
F4	UC4
F23	"UC27 UC8"
F45	UC35
F27	UC2
F7	UC42
F37	UC24
F44	"UC42 UC40"
F9	UC4
F31	UC16
F2	UC4
F10	UC4
F34	UC35

(b) List for spurious trace-links

Table 13: Misclassification for *dataset*-1 with $\geq$0.25 filtering

A.3 At least 0.67max

Table 14 shows the list of undetected trace-links and spurious trace-links.

Requirement id A	*Links* V
F8	"UC22 UC21 UC23"
F7	"UC23 UC21 UC22"
F10	UC14
F13	UC33
F6	"UC21 UC22 UC23"
F9	UC15
F14	UC32
F16	UC31
F25	UC18
F4	UC7

(a) List for undetected trace-links

Requirement id A	*Links* V
F28	"UC8 UC25"
F43	"UC14 UC18 UC7"
F12	"UC17 UC29 UC16 UC25"
F25	"UC2 UC8"
F11	UC4
F5	"UC7 UC4"
F6	"UC17 UC42"
F14	"UC17 UC7 UC15 UC16"
F38	UC24
F40	"UC24 UC7"
F13	"UC16 UC7 UC17 UC15"
F42	"UC27 UC8 UC2"
F36	UC42
F22	UC27
F39	UC42
F4	UC4
F21	"UC25 UC27"
F17	"UC27 UC25 UC24 UC8"
F23	UC27
F9	"UC2 UC4"
F32	UC25
F19	UC42
F46	"UC4 UC24 UC1"
F31	"UC15 UC16"
F45	UC35
F27	UC2
F8	UC42
F16	UC7
F7	UC42
F44	"UC42 UC40"
F37	UC24
F2	UC4
F10	UC4
F34	UC35

(b) List for spurious trace-links

Table 14: Misclassification for *dataset*-1 with $\geq$0.67max filtering

A.4 At least 0.67max with UB=2

Table 15 shows the list of undetected trace-links and spurious trace-links.

Requirement id A	*Links* V
F25	UC18
F6	"UC21 UC22 UC23"
F13	UC33
F4	UC7
F5	UC2
F7	"UC21 UC22 UC23"
F10	UC14
F8	"UC21 UC22 UC23"
F9	UC15
F14	UC32
F16	UC31

(a) List for undetected trace-links

Requirement id A	*Links* V
F25	"UC8 UC2"
F2	UC4
F7	UC42
F46	"UC4 UC1"
F13	"UC17 UC16"
F9	"UC2 UC4"
F11	UC4
F44	UC40
F31	"UC16 UC15"
F8	UC42
F45	UC35
F4	UC4
F14	"UC16 UC17"
F42	"UC8 UC27"
F43	"UC7 UC18"
F22	UC27
F40	"UC24 UC7"
F36	UC42
F21	"UC25 UC27"
F17	"UC25 UC27"
F10	UC4
F39	UC42
F6	"UC17 UC42"
F27	UC2
F34	UC35
F28	UC8
F12	"UC16 UC25"
F19	UC42
F38	UC24
F5	"UC4 UC7"
F37	UC24
F23	UC27
F32	UC25
F16	UC7

(b) List for spurious trace-links

Table 15: Misclassification for *dataset*-1 with $\geq$0.67max and UB=2 filtering

B Misclassification for Dataset-2

B.1 >0 filtering

Table 16 shows the list of undetected trace-links.

Requirement id A	*Links* V
F8	" UC21 UC16 UC15 UC11 UC14 UC22 UC13"
F12	UC15
F79	" UC17 UC9"
F74	UC7
F73	UC7
F71	UC9
F6	UC5
F30	UC17
F80	"UC9 UC17"
F55	UC1
F9	UC4
F36	UC15
F78	UC17
F54	UC1
F59	UC4
F47	UC1
F53	UC1
F34	UC13
F70	UC7
F10	UC4
F18	UC17
F23	UC23
F62	UC15

Table 16: List for undetected trace-links for *dataset*-2 with >0 filtering

Since the number of spuriously detected trace-links is too large and we cannot present these links by table within a page, we format them as follows:

- F72 UC15, UC14, UC12, UC17, UC10
- F17 UC14, UC12, UC4, UC8, UC17, UC6, UC18, UC11, UC7, UC23, UC5, UC13, UC15
- F40 UC15, UC6, UC8, UC20, UC22, UC14, UC18, UC9, UC11, UC4, UC17, UC21, UC13, UC19, UC10, UC23, UC7, UC12, UC5, UC16
- F60 UC20, UC18, UC10, UC12, UC17, UC19, UC3, UC21, UC11
- F77 UC8, UC1, UC6, UC16, UC7, UC13, UC11, UC21, UC12, UC14, UC3, UC23, UC18, UC17, UC10, UC4, UC19, UC15, UC5, UC20, UC22
- F36 UC23, UC17, UC21, UC11, UC10, UC18, UC12, UC3, UC20, UC1
- F54 UC12, UC10, UC17, UC3, UC11, UC20
- F48 UC14, UC11, UC4, UC16, UC20, UC18, UC15, UC12, UC6, UC22, UC10, UC7, UC17, UC8, UC23, UC21, UC3, UC19, UC13, UC9, UC5
- F13 UC12, UC13
- F8 UC21, UC20, UC22, UC9, UC5, UC6, UC14, UC11, UC19, UC12, UC15, UC18, UC7, UC10, UC17, UC4, UC13, UC23, UC16, UC3

- F46 UC17, UC18, UC11, UC20, UC1, UC23, UC3, UC21, UC10, UC12
- F66 UC9, UC8, UC16, UC12, UC13, UC15, UC17, UC14, UC10
- F21 UC20, UC6, UC9, UC19, UC8, UC7, UC3, UC21, UC10, UC13, UC12, UC17, UC22, UC18, UC11, UC5, UC14, UC15, UC16, UC4
- F7 UC10, UC22, UC21, UC18, UC20, UC12, UC11, UC4, UC7, UC8, UC19, UC23, UC3, UC14, UC13, UC17, UC16, UC9, UC15, UC6
- F49 UC8, UC21, UC13, UC12
- F19 UC11, UC20, UC18, UC17, UC10, UC12, UC23, UC1, UC3
- F15 UC17, UC7, UC12, UC8, UC14, UC10, UC15
- F55 UC11, UC12, UC10, UC20, UC17, UC3
- F22 UC19, UC17, UC6, UC16, UC4, UC21, UC23, UC12, UC9, UC15, UC22, UC5, UC8, UC20, UC10, UC11, UC7, UC18, UC13, UC14
- F12 UC21, UC20, UC17, UC23, UC3, UC12, UC18, UC11, UC10, UC1
- F25 UC10, UC20, UC16, UC8, UC5, UC14, UC12, UC19, UC7, UC11, UC3, UC1, UC21, UC4, UC9, UC18, UC23, UC17, UC22, UC6, UC13
- F52 UC10, UC21, UC20, UC18, UC17, UC11, UC23, UC3, UC12
- F44 UC1, UC8, UC4, UC12, UC5, UC17, UC11, UC18, UC20, UC3, UC10, UC6
- F68 UC9, UC17, UC15, UC14, UC12, UC16, UC10
- F32 UC20, UC13, UC21, UC9, UC8, UC19, UC11, UC12, UC16, UC22, UC14, UC10, UC6, UC3, UC17, UC15, UC18, UC5, UC4
- F30 UC13, UC7, UC12, UC8, UC16, UC9
- F18 UC16, UC9, UC7
- F62 UC1, UC10, UC20, UC23, UC21, UC12, UC3, UC17, UC18, UC11
- F3 UC7, UC10, UC17, UC15, UC19, UC4, UC3, UC20, UC14, UC1, UC21, UC8, UC9, UC22, UC6, UC11, UC23, UC18, UC13, UC16, UC12
- F26 UC5, UC20, UC13, UC15, UC6, UC17, UC18, UC8, UC12, UC10, UC7, UC21, UC11, UC14, UC4, UC1, UC23, UC19, UC16, UC22, UC9
- F67 UC8, UC15, UC17, UC14, UC9, UC16, UC7, UC13, UC12
- F76 UC8, UC9, UC13, UC3, UC12, UC11, UC5, UC10, UC15, UC22, UC4, UC17, UC19, UC20, UC21, UC16, UC6, UC14, UC18
- F38 UC23, UC3, UC6, UC22, UC5, UC21, UC14, UC16, UC20, UC15, UC17, UC13, UC12, UC9, UC18, UC19, UC10, UC8
- F11 UC10, UC21, UC5, UC6, UC15, UC12, UC7, UC23, UC9, UC19, UC1, UC22, UC16, UC14, UC17, UC13, UC3, UC20, UC8, UC11, UC4
- F53 UC10, UC17, UC3, UC21, UC11, UC20, UC12
- F27 UC7, UC17, UC11, UC9, UC8, UC16, UC4, UC13, UC15, UC1, UC22, UC12, UC10, UC5, UC3, UC19, UC14, UC6, UC18, UC20
- F35 UC20, UC17, UC7, UC12, UC23, UC3, UC14, UC18, UC16, UC9, UC1, UC11, UC10, UC21
- F23 UC10, UC19, UC12, UC3, UC8, UC13, UC18, UC21, UC20, UC6, UC15, UC11, UC16, UC9, UC5, UC17, UC7, UC14, UC4, UC23
- F29 UC11, UC3, UC17, UC12, UC20, UC1, UC10, UC18, UC19
- F2 UC3, UC11, UC9, UC20, UC8, UC22, UC17, UC21, UC4, UC6, UC13, UC12, UC15, UC19, UC10, UC18, UC16, UC14, UC7
- F28 UC5, UC17, UC6, UC12, UC18, UC8, UC10, UC23
- F31 UC14, UC21, UC8, UC20, UC19, UC4, UC13, UC15, UC10, UC7, UC3, UC16, UC5, UC6, UC12, UC18, UC9, UC22, UC17
- F1 UC23, UC4, UC3, UC18, UC15, UC22, UC13, UC17, UC5, UC16, UC10, UC11, UC9, UC12, UC14, UC21, UC8, UC19, UC7, UC20, UC6
- F5 UC17, UC8, UC15, UC9, UC10, UC3, UC6, UC11, UC13, UC20, UC21, UC12, UC7, UC4, UC19, UC22, UC14, UC18, UC16

- F69 UC16, UC17, UC10, UC14, UC12, UC8, UC15, UC9, UC13
- F42 UC11, UC3, UC10, UC4, UC21, UC23, UC20, UC18, UC22, UC1, UC17
- F63 UC10, UC16, UC18, UC17, UC8, UC4, UC23, UC7, UC6, UC5, UC15, UC12, UC14, UC19, UC1
- F24 UC8, UC1, UC3, UC5, UC18, UC7, UC10, UC9, UC13, UC6, UC17, UC16, UC11, UC14, UC23, UC19, UC20, UC15, UC22, UC4, UC21
- F37 UC19, UC1, UC12, UC17, UC10, UC7, UC15, UC14
- F78 UC9, UC15, UC7, UC16, UC13
- F39 UC4, UC23, UC12, UC18, UC17, UC6, UC5, UC10, UC8
- F33 UC15, UC12, UC14, UC10, UC17
- F45 UC20, UC6, UC4, UC11, UC10, UC1, UC3, UC18, UC5, UC17, UC19, UC8, UC12, UC23
- F58 UC18, UC10, UC1, UC12, UC17, UC3, UC23, UC11
- F20 UC10, UC6, UC4, UC23, UC8, UC17, UC5, UC12, UC18
- F64 UC14, UC15, UC17, UC1, UC16, UC7, UC10, UC12, UC19
- F43 UC6, UC1, UC11, UC13, UC20, UC17, UC21, UC4, UC16, UC8, UC19, UC12, UC18, UC3, UC14, UC10, UC23, UC22, UC9, UC7, UC5
- F14 UC14, UC12, UC17, UC7, UC10
- F61 UC10, UC20, UC3, UC21, UC11, UC17, UC18, UC23, UC12
- F75 UC6, UC23, UC19, UC1, UC5
- F9 UC12, UC21, UC20, UC3, UC1, UC10, UC11, UC18, UC17, UC23
- F34 UC7, UC15, UC14, UC10, UC17, UC13, UC12
- F41 UC19, UC22, UC7, UC3
- F51 UC20, UC11, UC10, UC17, UC3, UC18, UC12, UC21
- F79 UC1, UC19
- F57 UC19
- F59 UC21, UC11, UC19, UC3, UC17, UC12, UC10, UC20, UC18, UC1
- F16 UC18, UC17, UC7, UC10, UC20, UC1, UC21, UC12, UC3, UC11, UC14, UC23
- F70 UC19, UC1
- F65 UC9, UC16
- F50 UC21
- F71 UC19, UC1
- F80 UC19, UC1
- F56 UC21

B.2 At least 0.25

Since the number of undetected trace-links is too large and we cannot present these links by table within a page, we format them as follows:

- F8 UC21, UC16, UC11, UC8, UC15, UC14, UC22, UC13
- F61 UC1
- F12 UC15
- F74 UC7
- F73 UC7
- F51 UC1
- F16 UC15
- F64 UC9
- F5 UC5
- F52 UC1
- F49 UC1
- F55 UC1
- F14 UC15
- F63 UC9
- F15 UC13
- F78 UC17
- F60 UC1
- F54 UC1
- F59 UC4
- F47 UC1
- F11 UC18
- F10 UC4
- F18 UC17
- F42 UC12
- F28 UC4
- F62 UC15
- F34 UC8, UC13
- F20 UC19
- F40 UC3
- F26 UC3
- F79 UC17, UC9
- F77 UC9
- F48 UC1
- F7 UC5
- F24 UC12
- F13 UC8
- F71 UC9
- F44 UC23
- F6 UC5
- F25 UC15
- F30 UC17
- F21 UC23
- F80 UC9, UC17
- F9 UC4
- F75 UC7
- F36 UC15
- F17 UC10
- F72 UC7

- F35 UC15
- F53 UC1
- F33 UC7
- F2 UC5
- F50 UC1
- F70 UC7
- F39 UC19
- F68 UC7
- F43 UC15
- F23 UC23
- F41 UC23
- F3 UC5

Table 17 shows the list of spurious trace-links.

Requirement id A	*Links* V
F26	UC20
F31	"UC15 UC13"
F78	"UC16 UC13 UC9 UC15"
F18	"UC9 UC16"
F39	"UC6 UC5"
F20	"UC6 UC5"
F23	UC3
F17	"UC15 UC13"
F22	UC23
F35	"UC9 UC16"
F77	UC16
F64	UC16
F46	UC1
F42	UC4

Table 17: List for spurious trace-links for *dataset*-2 with >0.25 filtering

B.3 At least 0.67max

Table 18 shows the list of undetected trace-links.

Since the number of spurious trace-links is too large and we cannot present these links by table within a page, we format them as follows:

- F8 UC21, UC16, UC11, UC8, UC15, UC14, UC22, UC13
- F61 UC1
- F12 UC15
- F74 UC7
- F73 UC7
- F51 UC1
- F16 UC15
- F64 UC9
- F5 UC5
- F52 UC1
- F49 UC1
- F55 UC1
- F14 UC15
- F63 UC9
- F15 UC13
- F78 UC17
- F60 UC1
- F54 UC1
- F59 UC4
- F47 UC1
- F11 UC18
- F10 UC4
- F18 UC17
- F42 UC12
- F28 UC4
- F62 UC15
- F34 UC8, UC13
- F20 UC19
- F40 UC3
- F26 UC3
- F79 UC17, UC9
- F77 UC9
- F48 UC1
- F7 UC5
- F24 UC12
- F13 UC8
- F71 UC9
- F44 UC23
- F6 UC5
- F25 UC15
- F30 UC17
- F21 UC23
- F80 UC9, UC17
- F9 UC4
- F75 UC7
- F36 UC15

- F17 UC10
- F72 UC7
- F35 UC15
- F53 UC1
- F33 UC7
- F2 UC5
- F50 UC1
- F70 UC7
- F39 UC19
- F68 UC7
- F43 UC15
- F23 UC23
- F41 UC23
- F3 UC5

B.4 At least 0.67max with UB=2

Table 19 shows the list of undetected trace-links

Since the number of spurious trace-links is too large and we cannot present these links by table within a page, we format them as follows:

- F78 UC16
- F55 UC20, UC10
- F70 UC1
- F2 UC14, UC15
- F28 UC6, UC5
- F40 UC23
- F11 UC20, UC3
- F71 UC1
- F6 UC23, UC22
- F9 UC3, UC20
- F64 UC16
- F77 UC16
- F36 UC3, UC20
- F79 UC1
- F10 UC23, UC22
- F54 UC10, UC20
- F72 UC15, UC14
- F73 UC22, UC23
- F25 UC20, UC3
- F42 UC4, UC21
- F48 UC20, UC11
- F17 UC15, UC13
- F16 UC20, UC10
- F20 UC5, UC6
- F15 UC8
- F74 UC22, UC23
- F44 UC5
- F5 UC6
- F34 UC13
- F80 UC1
- F62 UC20, UC3
- F12 UC20, UC3
- F30 UC7, UC16
- F63 UC1
- F8 UC11, UC10
- F13 UC13
- F18 UC16, UC9
- F7 UC6
- F21 UC6, UC5
- F35 UC16, UC9
- F47 UC22, UC23
- F37 UC1
- F53 UC21
- F3 UC6
- F33 UC14, UC15
- F61 UC20, UC3

- F75 UC19
- F68 UC16
- F14 UC14
- F26 UC20
- F60 UC21
- F43 UC20, UC3
- F24 UC20, UC3
- F49 UC21
- F23 UC3
- F39 UC5, UC6
- F1 UC6, UC5
- F46 UC1
- F59 UC21

Requirement id A	*Links* V
F8	" UC21 UC16 UC15 UC11 UC14 UC8 UC22 UC13"
F61	UC1
F12	UC15
F26	UC3
F79	" UC17 UC9"
F48	UC1
F24	UC12
F71	UC9
F16	UC15
F25	UC15
F30	UC17
F21	UC23
F80	"UC9 UC17"
F9	UC4
F55	UC1
F36	UC15
F17	UC10
F72	UC7
F63	UC9
F35	UC15
F78	UC17
F60	UC1
F54	UC1
F59	UC4
F53	UC1
F34	UC13
F33	UC7
F11	UC18
F2	UC5
F70	UC7
F18	UC17
F39	UC19
F43	UC15
F42	UC12
F23	UC23
F62	UC15
F20	UC19

Table 18: List for undetected trace-links for *dataset*-2 with >0.67max filtering

Requirement id A	*Links* V
F63	UC9
F18	UC17
F8	" UC22 UC8 UC13 UC15 UC14 UC16 UC21 UC11"
F70	UC7
F24	UC12
F53	UC1
F74	UC7
F35	UC15
F54	UC1
F30	UC17
F36	UC15
F62	UC15
F2	UC5
F42	UC12
F16	UC15
F73	UC7
F61	UC1
F80	" UC17 UC9"
F9	UC4
F60	UC1
F26	UC3
F33	UC7
F34	UC13
F47	UC1
F10	UC4
F12	UC15
F28	UC4
F23	UC23
F17	UC10
F59	UC4
F72	UC7
F21	UC23
F11	UC18
F39	UC19
F78	UC17
F48	UC1
F55	UC1
F79	" UC17 UC9"
F20	UC19
F25	UC15
F43	UC15
F71	UC9
F6	UC5

Table 19: List for undetected trace-links for *dataset*-2 with ≥0.67max and UB=2 filtering