Requirements Traceability

Denis Shehu
1232758

Richard Wellens
1246569

I. Introduction

In this assignment we will implement three methods to detect trace-links between high level and low level requirements. Additionally we will come up with one custom technique that outperforms the three given ones. The main benefits of requirements traceability are the ability to validate whether all requirements are satisfied and that it allows impact analysis to be performed on proposed changes. These are valuable insights during development as both the requirements and implementation evolve over time. In the past trace link detection was done manually, which is a long and arduous process. This is especially the case for large projects where it can take a single person around 12 hours to perform the trace. The time cost of this method makes it unsuitable for a continuous development environment and is usually used after (a part of) the software has been delivered. Trace-link detection tools can speed up the process of performing the trace by finding trace-links and having an expert check the results. In a development environment a trace-link detection tool can be used to suggest links between new additions and the existing requirements or between commits and existing issues. This continuous updating of the trace-links will provide the benefits described above while keeping the time overhead manageable.
In Section 2 we will discuss the implementation details of the trace-link detection methods. Section 3 will describe the datasets used to evaluate their performance followed by the evaluation in Section 4. Section 5 contains the discussion and Section 6 the related work. Finally Section 7 contains our conclusions.

II. Trace-link detection tool

A. Implementation details

The base-line methods use the same implementation for preprocessing, generating the vector representation and generating the similarity matrix. They differ only in the fact that they use different threshold values to determine whether two requirements are related. Preprocessing is done in the following way:
Once the dataset is read from the csv file the requirement ID and original text are saved as individual requirements in requirement objects. When preprocessing starts we remove stop words by comparing them to a list of stop words provided by the SMILE library [4]. We used the google list (from the available lists in the library), since we found that it performed the best. On the remaining words we apply the Lancaster stemming-algorithm whose implementation is also provided by the SMILE library. We chose the Lancaster algorithm since we found that it slightly outperformed the Porter algorithm (which was part of the same library) on our datasets. However the Lancaster algorithm might not be preferable in cases where more distinction between concepts is required, as it is more aggressive in its stemming than the Porter algorithm. Once the preprocessing is complete the strings are tokenized and saved in a dataset class.
Using the preprocessed dataset we generate a vector representation for each requirement. To do this we first construct a master vocabulary containing all unique words in the preprocessed dataset, this vocabulary has size N. We then find the frequency of each word in each requirement. Using this we compute the vector representation $\langle w_1, \dots, w_n \rangle$ such that $w_i = 0$ if word i in the master vocabulary does not occur in the requirement. Otherwise $w_i = tf \cdot idf$, where tf is the frequency of the ith word in the requirement and $idf = log_2(\frac{n}{d})$ where n is the total number of requirements and d is the number of requirements containing the ith word. The resulting vector representations are then saved in their respective requirement object. The next step is to compute the similarity matrix. To compute the similarity we implemented the cosine similarity

$$sim(X,Y) = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \qquad (1)$$

as it is easy to implement and mandated by the assignment for the base-line methods.
Finally we determine the links between the requirements based on their similarity values. The first and second base-line methods use a threshold value of 0 and 0.25 respectively. For the third method we find for every high-level requirement h the low-level requirement l with the highest similarity. We then create a link between h and any low level requirement l' when $sim(h, l') \geq 0.67 \cdot sim(h, l)$. Our alternative to these methods is used in our custom method and is described below.

B. Custom technique

For our custom technique we used the same preprocessing, vector representation and similarity calculation methods as the base-line methods described above. We did this because we only wanted to change one factor while improving the base-line methods, so it is completely clear what factor contributes to the improvement. We decided to improve the way the links are generated given the similarity values for all requirements. We ended up with the following method: For each high level requirement h we find the low level requirement l with the highest similarity. We then compute the median and standard deviation of these values and use that to compute our threshold value t: $t = median + 0.1 * std$. The reasoning behind this is that it is an improvement on the third base-line method where the threshold value for a link between h and l is based on the highest similarity of any l' with that same h. This means that for any h there is always at least one link (the one with the highest similarity), which is not always correct. By computing the median of the highest similarity values for every h, we get a better view of the 'bigger picture' and reduce the chance of linking high level requirements that should not have any links. The idea behind using the standard deviation rather than a constant fraction of the median is that the standard deviation describes the overall distribution of these values better. Our testing also affirms this as using the standard deviation performed slightly better than a constant fraction. The constant value of 0.1 was found by trying multiple values and comparing performance.

C. Tool usage

We were unable to deliver a functional tool using docker as the assignment required. This is due to the fact that we lacked experience in docker. We created a standard Java project and assumed we could get it to work with docker after it was done. As can be seen from the results we underestimated how much time this would cost and could not provide a functional docker project. In the future we will get a skeleton project to work in the correct environment first to avoid this issue.
As the source code is provided with the assignment, it can still be run as a Java project.
The input for choosing the trace-link generation technique is read by using the Scanner class.

III. Datasets

While the tool was being developed, two groups from the Waterloo dataset were used to verify its correctness and accuracy. Both these groups consisted of three files in the `.csv` format, namely: `high.csv`, `low.csv`, and `links.csv`.

The first two store respectively the list of high and low level requirements. Each of these requirements is represented by a `string` value named `id`, corresponding to the ID of the requirement, and another `string` value named `text`, corresponding to the actual requirement's text.

The third file (i.e. `links.csv`) stores a list of the links between the high and low level requirements that were computed manually. Each link is represented by two `string` values (namely `id` and `links`), the first corresponds to the ID of a high level requirement and the second corresponds to a concatenation of the IDs of all the low level requirements link to that high level requirement, separated by commas. As an example, if high level requirement `F1` is linked to low level requirements `UC1`, `UC2`, and `UC3`, the corresponding instance in the `links.csv` file would be `F1: ``UC1,UC2,UC3''`.

The first group of the Waterloo dataset (`dataset-1`) had 49 high level requirements and 26 low level requirements. The second group (`dataset-2`) had 80 high level requirements and 22 low level requirements.

IV. Evaluation

As mentioned previously, the tool has four techniques for trace-link detection, three of which are baseline techniques and the fourth is a custom proposed technique. Each of them was evaluated and the results are presented in the following four subsections.

A. The first baseline technique

As introduced previously, the first baseline technique predicts the links by whether or not their corresponding similarity values are larger than 0. Tables I and II present the confusion matrices corresponding to `dataset-1` and `dataset-2` respectively.

What can be concluded from these tables are the following:

1) If the similarity value corresponding to a link is equal to 0, then that link has a high change of not being manually identified. This conclusion is derived in the following manner: the column corresponding to the similarity values equal to 0 is *"Trace-link not predicted by the tool"*. Out of 265 links, only two of them are not identified manually, so they are false negatives. As a result, the tool can predict with a high probability that a link is not manually identified. This is also reflected on the high recall value that the confusion matrix of Table I has, i.e. $\frac{37}{39} \approx 0.95$.
2) The issue with this method is when predicting a trace-link, since it is not sufficient to assume that only because the similarity value is above 0 this link is manually identified. Those links that have quite small similarity values make the cut and are predicted by the tool, which increases the number of false positives (970 for `dataset-1`). As a result, the precision value is close to 0 (0.0367 for the first dataset and 0.0648 for the second).

	Trace-link predicted by the tool	Trace-link not predicted by the tool
Trace-link identified manually	37	2
Trace-link not identified manually	970	265

TABLE I
THE CONFUSION MATRIX OF `DATASET-1`.

	Trace-link predicted by the tool	Trace-link not predicted by the tool
Trace-link identified manually	57	31
Trace-link not identified manually	822	1570

TABLE II
THE CONFUSION MATRIX OF `DATASET-2`.

For `dataset-1` there are only two links that are identified manually, but are not predicted by the tool, namely the links between high level requirement `F6` and low level requirements `UC22` and `UC23`. Even though they are manually identified, these links have a similarity value of 0.

Those links that are identified manually, but are not predicted by the tool for `dataset-1` are in the order of 970, however examples like the links between `F1` and `UC4`, `UC6`, whose respective similarity values are 0.0031 and 0.0026, show the downfalls of the first baseline method.

For the second dataset, the number of links that are identified manually, but are not predicted by the tool (i.e. the number of false negatives) is higher than for the first dataset, but still significantly lower than the number of true negatives. All these misclassifications are the following: (in order to save space, a link `Fx: UCy` is abbreviated as x: y)

6: 5
8: 11, 13, 14, 15, 16, 21, 22
9: 4
10: 4
12: 15
18: 17
23: 23
30: 17
34: 13
36: 15
47: 1
53: 1
54: 1
55: 1
59: 4
62: 15
70: 7
71: 9
73: 7
74: 7
78: 17
79: 9, 17
80: 9, 17

Again, the number of false positives is too high to be presented in here, but a meaningful example is the link between `F2` and `UC4` which has a similarity value of 0.0174.

B. The second baseline technique

The second baseline technique is an improved version of the first, i.e. instead of predicting links based on their similarity value being equal to 0 or not, the threshold is increased to 0.25. This way, the argument goes, links that are manually identified will probably have a similarity value higher than 0.25, and those that are not manually identified will ideally have a similarity value of 0, or some other value less than 0.25.

The confusion matrices corresponding to the first and second datasets for this method are shown in Tables III and IV respectively. Even though the performance of the second method is significantly better than the first (the F-measure has increased approximately six folds), this method still suffers from the same downfalls as the first, but the impact of the most influential one is reduced. These downfalls are the following:

1) In the first method, all links with similarity values equal to 0 were not predicted by the tool, and still there were two false negatives. With the second method, as mentioned previously, this threshold was increased to 0.25, so it is a logical consequence that the number of false negatives would increase as well. For `dataset-1` the recall value of the first method was ≈ 0.95 and for this method it is ≈ 0.51, reflecting the result that the increase of the threshold had on performance, it increased the number of false negatives.
2) Even though this method has a lower recall than the first, its value for precision is the reason why the F-measure is much higher. The most influential downfall of the previous method was that links with a low similarity value, but still not equal to 0, would be predicted. The second method eliminates these cases by increasing the threshold and as a result, for `dataset-1` the number of false positives drops significantly from 970 to only 33.

	Trace-link predicted by the tool	Trace-link not predicted by the tool
Trace-link identified manually	20	19
Trace-link not identified manually	33	1202

TABLE III
THE CONFUSION MATRIX OF `DATASET-1`.

	Trace-link predicted by the tool	Trace-link not predicted by the tool
Trace-link identified manually	19	69
Trace-link not identified manually	21	2291

TABLE IV
THE CONFUSION MATRIX OF `DATASET-2`.

For `dataset-1`, the links that are identified manually, but are not predicted by the tool are the following:

1: 1
4: 7
6: 21, 22, 23
7: 21, 22, 23
8: 21, 22, 23
9: 15
10: 14
13: 33
14: 32
15: 29
16: 31
26: 25
32: 24

An interesting case of these misclassifications is the link between `F15` and `UC29`. The similarity value is approximately 0.23, so just a bit below 0.25. This link was predicted in the first method, but because of the threshold's increase it became a false negative in this method. This also exposes another issue with this and the previous method: their threshold is static and does not reflect the distribution of the similarity values.

The links of `dataset-1` that are not identified manually, but are predicted by the tool are the following:

2: 4
4: 4
5: 4, 7
7: 42
9: 4
10: 4
11: 4, 7, 2
23: 27, 8
24: 7, 2, 27, 18
25: 2, 8
27: 2
28: 8
29: 17
30: 25, 16
31: 16
33: 1
34: 35
35: 42
37: 24
42: 27, 8
44: 42, 40
48: 16

High level requirement F2 has a similarity value of 0.35 with UC6, and indeed their link is identified manually, but it also has a similarity value of 0.31 with UC4, even though their link is not manually identified. This reflects an issue with the process of computing the representation vectors, but this will be discussed later in this paper.

The same situation is seen for dataset-2, its false negatives are the following:

2: 5
3: 5
5: 5
6: 5
7: 5
8: 8, 11, 13, 14, 15, 16, 21, 22
9: 4
10: 4
11: 18
12: 15
13: 8
14: 15
15: 13
16: 15
17: 10
18: 17
20: 19
21: 23
23: 23
24: 12
25: 15
26: 3
28: 4
30: 17
33: 7
34: 8, 13
35: 15
36: 15
39: 19
40: 3
42: 12
43: 15
44: 23
47: 1
48: 1
49: 1
50: 1
51: 1
53: 1
54: 1
55: 1
59: 4
60: 1
61: 1
62: 15
63: 9
64: 9
68: 7
70: 7
71: 9
72: 7
73: 7
74: 7
75: 7
76: 7
77: 9
78: 17
79: 9, 17
80: 9, 17

The tool has missed the link between low level requirement UC5 and several high level requirements, like F2, F3, F5, etc (with similarity values of 0.02, 0.15, 0.02 respectively). Again, this points out an issue with the vector representation computation and UC5 is not the only case, the same situation applies to requirements like UC1 and UC7 among others.

The false positives for dataset-2 also reflect this issue, like the link between F22 and UC23 that has a similarity value of 0.3, even though the link is not identified manually. The complete list of the false positives of dataset-2 is the following:

17: 15, 13
18: 9, 16
20: 5, 6
22: 23
23: 3
26: 20
27: 22
31: 15, 13
35: 16
39: 5
41: 22
46: 1
64: 16

C. The third baseline technique

As it was briefly mentioned before, the first two techniques are in their core the same: there is a predefined value, a threshold, and that decides whether a link is predicted or not. And the issue with this approach is that the distribution of the similarity values is not taken into account. In the most extreme case, the same threshold could be higher than any similarity value of the dataset so no links are predicted, and for another dataset it could be lower than all the similarity values, therefore every link is predicted.

The third baseline technique tries to resolve this issue, by setting a different threshold for each high level requirement, which is based on the similarity value of the most similar low level requirement.

Even though this technique tries to solve some of the downfalls of the previous two, it still has its own, namely the following:

1) Since we are basing the threshold of every high level requirement on the highest similarity value it has with any of the low level requirements, by default the link between the high and the most similar low level requirement will always be predicted by the tool. As a result, for every high level requirement there will always be at least one low level requirement linked to it, and this is an issue since there are high level requirements that are not manually linked with any low level requirements.
2) The solution that the third technique provides for the downfalls of the previous two techniques is a bit aggressive to the point that it causes problems, but in the other side of the spectrum. In other words, the threshold is so "elastic" it does not take into account the general structure of the dataset. The highest similarity value for a particular high level requirement could be so low that it probably needs to be neglected, so we could end up in the situation described in point 1), but the tool does not "see" that because it does not take the whole dataset into account.
3) Another issue is the static value by which the highest similarity value is multiplied. For a particular high level requirement, 0.67 could be a pretty good fit, but for another one the threshold might need to be multiplied by a lower/higher value to achieve a best performance.

Tables V and VI give the confusion matrices for the first and second datasets respectively using the third baseline technique.

	Trace-link predicted by the tool	Trace-link not predicted by the tool
Trace-link identified manually	24	15
Trace-link not identified manually	56	1179

TABLE V
THE CONFUSION MATRIX OF DATASET-1.

	Trace-link predicted by the tool	Trace-link not predicted by the tool
Trace-link identified manually	39	49
Trace-link not identified manually	209	2183

TABLE VI
THE CONFUSION MATRIX OF DATASET-2.

The false negatives of `dataset-1` are the following:

4: 7	8: 21, 22, 23	13: 33
6: 21, 22, 23	9: 15	14: 32
7: 21, 22, 23	10: 14	16: 31

A representative of these links is the one between `F4` and `UC7`. This link demonstrates the staticity of the 0.67 coefficient because for the high level requirement `F4` the threshold his higher than it should have been. The similarity between these two requirements is approximately 0.28, and this link for instance would have been predicted in the first two techniques.

A great representative of the false positives for this technique is the class of links of high level requirements that do not have any manually identified links (this situation corresponds to point 1) of the downfalls of this technique). Such examples include requirements like `F11`, `F12`, and `F17` among others. The full list of false positives for `dataset-1` is the following:

2: 4	12: 16, 35	25: 2, 8	39: 42, 40
4: 4	13: 35	27: 2	40: 23
5: 4, 7	14: 7, 2, 16,	28: 8, 25	42: 27, 8
6: 36, 17, 42,	17	29: 17	43: 7, 14, 18
40	16: 6, 7	31: 15, 16	44: 42, 40
7: 42	17: 27, 25	32: 25	46: 1, 24
8: 42	19: 42	34: 35	
9: 4, 2	21: 27, 25	36: 42	
10: 4	22: 27, 8	37: 24	
11: 4	23: 27	38: 24	

The same situation as with the first dataset can be seen for the second. The false negatives are those links for which the 0.67 coefficient is to high, like the link between `F40` and `UC3` that has a similarity value of 0.12. This value might not seem like much but when compared with other similarity values like the ones `F40` has with `UC11`, `UC12`, `UC18` that do not go beyond 0.008, 0.12 is significantly higher. All the false negatives for `dataset-2` are the following:

2: 5	20: 19	39: 19	62: 15
7: 5	21: 23	40: 3	63: 9
8: 8, 11, 13,	23: 23	42: 12	70: 7
14, 15, 16,	24: 12	43: 15	71: 9
21, 22	25: 15	48: 1	72: 7
9: 4	26: 3	53: 1	73: 7
11: 18	30: 17	54: 1	78: 17
12: 15	33: 7	55: 1	79: 9, 17
16: 15	34: 13	59: 4	80: 9, 17
17: 10	35: 15	60: 1	
18: 17	36: 15	61: 1	

The list of false positives for `dataset-2` is so extensive it cannot be shown in here entirely. However, the following can be seen as fit representatives because all requirements `F1`, `F37`, and `F46` do not have any manually identified links.

```
F1: UC4, UC5, UC6
F37: UC1
F46: UC1
```

However, these are not the only false positives, and the reason why other false positives exist has to do with the way the vector representations are generated, but that it explained later in this paper.

D. The custom technique (DONE)

As it was mentioned on the previous subsection, the third technique always predicts at least one link for each high level requirement, and it fails to look at the "bigger picture". These downfalls are addressed on the final technique, the custom one. Tables VII and VIII present the confusion matrices of `dataset-1` and `dataset-2` respectively.

	Trace-link predicted by the tool	Trace-link not predicted by the tool
Trace-link identified manually	19	20
Trace-link not identified manually	15	1220

TABLE VII
THE CONFUSION MATRIX OF `DATASET-1`.

	Trace-link predicted by the tool	Trace-link not predicted by the tool
Trace-link identified manually	26	62
Trace-link not identified manually	42	2270

TABLE VIII
THE CONFUSION MATRIX OF `DATASET-2`.

This technique for `dataset-1` has the lowest number of false positives, and for `dataset-2` it has the second lowest, implying that the threshold has never had a better value to correctly discard a link as probably not being manually identified. However, there are predicted links whose similarity value is too high even though these links are not manually identified. A possible explanation could be the way the vector representation is computed (this is discussed in the Discussion section).

Such examples are the link between `F4` and `UC4` in `dataset-1` (the similarity value of this link is 0.588) and the link between `F46` and `UC1` in `dataset-2` (with similarity value of 0.56). The following lists present the false positives for `dataset-1` and `dataset-2` respectively.

`dataset-1`:

4: 4	24: 7, 2	31: 16	
5: 4	25: 2, 8	37: 24	
11: 4	27: 2	44: 42, 40	
23: 27	30: 25	48: 16	

`dataset-2`:

13: 13	27: 22	42: 4, 6	63: 1
17: 15, 13	31: 15, 13	43: 8, 9	64: 16
18: 9, 16	35: 9, 16	45: 5	75: 19, 1
20: 5, 6	38: 5	46: 1, 20	77: 16
22: 23, 22	39: 5, 6	56: 21	78: 15, 13,
23: 23, 3	40: 23	59: 21	9, 16
26: 1, 20	41: 22	60: 21	

When discussing the false negatives, there are links with low similarity values even though they are manually identified, thus they do not make the cut. Again, this could point out an issue with the vector representation generation and as mentioned previously that is addressed in the Discussion section.

Such false negatives could be the link between `F10` and `UC14` in `dataset-1` that has a similarity value of 0.087, or the link between `F2` and `UC5` in `dataset-2` with value 0.022. The full list of false negatives for both datasets are the following:

`dataset-1`:

1: 1	8: 21, 22, 23	14: 32	32: 24
4: 7	9: 15	15: 29	49: 42
6: 21, 22, 23	10: 14	16: 31	
7: 21, 22, 23	13: 33	26: 25	

`dataset-2`:

2: 5	16: 15	39: 19	61: 1
3: 5	17: 10	40: 3	62: 15
5: 5	18: 17	42: 12	63: 9
6: 5	21: 23	43: 15	68: 7
7: 5	23: 23	44: 23	70: 7
8: 8, 11, 13,	24: 12	47: 1	71: 9
14, 15, 16,	25: 15	48: 1	72: 7
21, 22	26: 3	49: 1	73: 7
9: 4	28: 4	51: 1	74: 7
10: 4	30: 17	53: 1	78: 17
11: 18	33: 7	54: 1	79: 9, 17
12: 15	34: 8, 13	55: 1	80: 9, 17
14: 15	35: 15	59: 4	
15: 13	36: 15	60: 1	

E. Techniques comparison

In this section, the differences between the the four implemented techniques will be discussed. Table IX compares the values of the recall, precision, and F-measure for the four techniques on `dataset-1` and the same comparison is presented on Table X but for `dataset-2`.

As it can be seen from Tables IX and X, the technique with the highest F-measure (which is seen as an overall performance measure) is the proposed one for both datasets. Thus, the custom technique generally performs better.

The custom technique also has the highest precision value for `dataset-1` and the second highest for `dataset-2`, as it was previously explained this implies that the threshold has never had a better value to correctly discard a link as probably not being manually identified.

Its recall value is lower when compared to the other techniques, which implies that the threshold is higher than necessary, so many links that are manually identified do not make the cut. As mentioned in Section II point B many values were tried for the threshold and the current one had the best performance. Both a higher and lower threshold performed worse, so again this can point out at the process of generating the vector representations.

	No filtering	≥ 0.25	$\geq 0.67 \cdot \max_{sim}$	Custom technique
Recall	0.949	0.513	0.615	0.487
Precision	0.037	0.377	0.3	0.559
F-measure	0.071	0.435	0.403	0.521

TABLE IX
THE COMPARISON OF THE FOUR TECHNIQUES USING `DATASET-1`.

	No filtering	≥ 0.25	$\geq 0.67 \cdot \max_{sim}$	Custom technique
Recall	0.648	0.216	0.443	0.295
Precision	0.065	0.475	0.157	0.382
F-measure	0.118	0.297	0.232	0.333

TABLE X
THE COMPARISON OF THE FOUR TECHNIQUES USING `DATASET-2`.

The following is a list that illutrates the different misclassifications of the four techniques:

1) The first technique (i.e. no filtering) is very conservative on not predicting a link (the similarity value must be exactly 0), but at the same time it is very liberal on predicting a link (a similarity value of slightly above 0 is sufficient). This causes the method to have the highest recall value, but the lowest precision. Thus, the method does really well on avoiding false negatives, but barely does anything on avoiding the false positives.
2) The second technique, as discussed previously, is an improvement on the first, meaning that it is less conservative on not predicting a link (the similarity value must be lower than 0.25) and it is less liberal on predicting a link (similarity value is at least 0.25). As a result, this technique produces more false negatives than the first, but significantly less false positives. The improvement on the number of false positives is far greater than the deterioration on the number of false negatives, thus the increase on precision is greater than the decrease on recall and as a result the overall performance is increased.
3) The third technique tries to improve the staticity of the threshold of the first two techniques. The result of

this seems to be a generally higher threshold, which improves the recall value when compared to the second technique. However, this increase decreases the precision (especially for the second dataset) to that point that the overall performance is worse than the one of the second technique. However, this technique suffers from the issue that every high level requirement is linked to at least one low level requirement.

4) The custom technique tries to avoid the last issue mentioned on the previous technique, but in the process it does not predict links that are manually identified because the similarity values of these links are pretty low. However, the precision of this technique is the highest, so lowering the threshold would not be smart. In the next section, the process of generating the vector representations is discussed as a cause of having low or high similarity values for links that need to be predicted or not respectively.

V. Discussion

It has been repeatedly mentioned throughout this paper that a reason why so many links that had to be predicted had a very low similarity value, or links that had to not be predicted had a very high similarity value, might have been caused by the way the vector representations are generated.

It has been demonstrated in many instances in the Evaluation section that for some links it did not matter where the threshold was placed, they would be classified incorrectly even though the performance of the technique was better than ever with the given threshold.

While developing the fourth technique, the highest similarity value for each high level requirement was plotted on 1D-space and the corresponding result for `dataset-1` is shown on Figure 1. The mean was equal to approximately 0.337 and the median was 0.309, which are quite low values. It was expected that the highest similarity values would form a cluster close to the value of 1.0, since there are so many similarity values that are above the mean or the median and still are not the highest value for their corresponding high level requirement.

The list of English stop words chosen was the one that performed the best when compared with other lists, but still the most fit list for a dataset depends a lot on the dataset itself.

Also, the used stemming algorithm was the best performing one, but as mentioned on Section II point A, the Lancaster's algorithm is aggressive in stemming.

This could have been the reasons why for so many links the similarity values did not reflect correctly their status on being or not predicted, which in turn lowered the overall performance of the techniques.

VI. Related work

A. Towards Semantically Guided Traceability

Liu *et al.* [1] describe a method to improve accuracy and recall of trace-link generation over traditional VSM by mining existing works in a specific domain to obtain frequently used

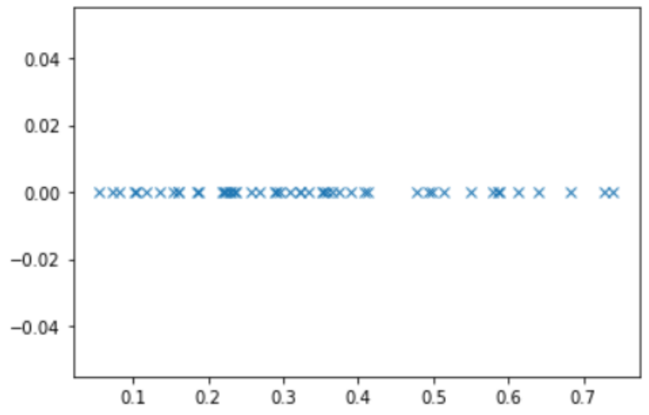

Fig. 1. The distribution of the highest similarity values per high level requirement.

concepts and their relations. Their method uses AutoPhrase [2] to extract concepts from existing work in a specific domain (e.g. web-search, white-papers, source code and academic literature) and the Stanford Parser [3] to identify relations between concepts like synonyms, acronyms, siblings and ancestor-descendants. On the resulting network they use fuzzy matching to find all unique non-redundant paths that connect two concepts and use a shortest path algorithm to find the best one. If the best path found is shorter then threshold value τ it is considered related, which will be used in the computation of the similarity.

In contrast our approach uses the similarity calculation from VSM but uses a different method to find the links between the requirements. These methods could be combined for even better performance, this might be a topic for further study. Liu *et al.* found that their method performed noticeably better than traditional VSM (F = 0.220 and 0.196 respectively), especially when they combined the two methods into a hybrid approach (F = 0.253). Our approach in comparison had a more significant gain in F-measure compared to traditional VSM, however our tests were done on a much smaller dataset and thus are not directly comparable.

B. Traceability in the Wild. Automatically Augmenting Incomplete Trace Links

Rath *et al.* [5] took a different approach to trace-link generation, as their goal was to find missing links between git commits and issues tracked by Jira in a development setting. This meant that they have existing data to work with, for developers usually link their commit to an issue. They identified 18 attributes that can be divided into two categories: process-related attributes, which capture stakeholder-related, temporal and structural characteristics, and textual similarity attributes, which are the attributes that capture textual and semantic association. Using these attributes they aim to train a supervised learning classifier, their candidate methods include Naive Bayes, J48 Decision Tree and Random Forest. The Random Forest classifier performed the best with a recall of

0.916 and precision of 0.173.
What is interesting about this approach compared to our own is that we only use textual similarity while they add additional process-related attributes to increase accuracy and recall. Also the use of machine learning might be of interest for future study, but due to time limitations this was beyond the scope of our assignment. Our method has a slightly higher f-measure than reported here, but the results are not directly comparable with ours since they use a completely different dataset.

VII. Conclusions

In this assignment we implemented three base-line methods for trace-link generation, analyzed their performance and introduced a fourth method that outperforms them. In our approach the threshold value for linking two requirements is based on the highest similarity values for all high level requirements rather than just the one involved in this particular link. Though it performs better than the base-line approach there are still many ways in which they can be improved. For example a thesaurus can be used to take into account synonyms, acronyms and similar words of different types (that are not caught by stemming). Another way to improve the accuracy is to take into account additional information, like temporal or structural characteristics as described in Rath *et al*. [5].

References

[1] Y. Liu, J. Lin, Q. Zeng, M. Jiang and J. Cleland-Huang, ”Towards Semantically Guided Traceability,” 2020 IEEE 28th International Requirements Engineering Conference (RE), Zurich, Switzerland, 2020, pp. 328-333, doi: 10.1109/RE48521.2020.00043.

[2] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han. Automated-phrase mining from massive text corpora.IEEE Trans. Knowl. DataEng., 30(10):1825–1837, 2018.

[3] M.-C. De Marneffe and C. D. Manning. Stanford typed dependencies-manual. Technical report, Technical report, Stanford University, 2008.

[4] H. Li, ”Statistical Machine Intelligence and Learning Engine”, Github Repository, https://github.com/haifengl/smile, 2020

[5] Rath, M., Rendall, J., Guo, J. L. C., Cleland-Huang, J., Mäder, P. (2018). Traceability in the wild. Proceedings of the 40th International Conference on Software Engineering - ICSE ’18. doi:10.1145/3180155.3180207