TRACEABILITY RECOVERY
-
Classroom Exercise

In order to perform this little exercitation it is required to read the following papers:

[1] The following are seminal papers in the field of IR-based Traceabiity Recovery: O. C. Z. Gotel, Anthony Finkelstein, "An analysis of the requirements traceability problem", In: Proceedings of the Proceedings of the First IEEE International Conference on Requirements Engineering (ICRE), 1994.

[2] Giovanni Capobianco, Andrea De Lucia, Rocco Oliveto, Annibale Panichella, Sebastiano Panichella, " On the role of the nouns in IR-based traceability recovery", In: Proceedings of the IEEE 17th International Conference on Program Comprehension (ICPC 2009).

These two papers represent the base literature-based Traceability recovery. Specifically, Traceability Recovery represents a key software maintenance activity in which software engineers extract the relationships among software artifacts. Extract and manage traceability between software artifacts is important for software development and maintenance: program comprehension, requirement tracing, impact analysis and software reuse. However, trace links between software artifacts is a practice not widely adopted by developers because represents a very time consuming task. For this reason, previous works proposed an approach to instantiate the problem of Traceability Recovery as a Information Retrieval problem. The conjecture is that most software artifacts contain text. Artifacts with high similarity probably describe similar concepts thus they are likely good candidates to be traced onto each other. Consistent use of domain terms in the software documents (e.g., programmers use meaningful identifiers) impact the results of IR-based Traceability Recovery approaches.

Figure 1. shows the typical process of an IR-based Traceability Recovery (Slides of the lecture related to “Reengineering I” available

in http://www.ifi.uzh.ch/seal/teaching/courses/softwareWartung-FS16.html explain more in detail this process).

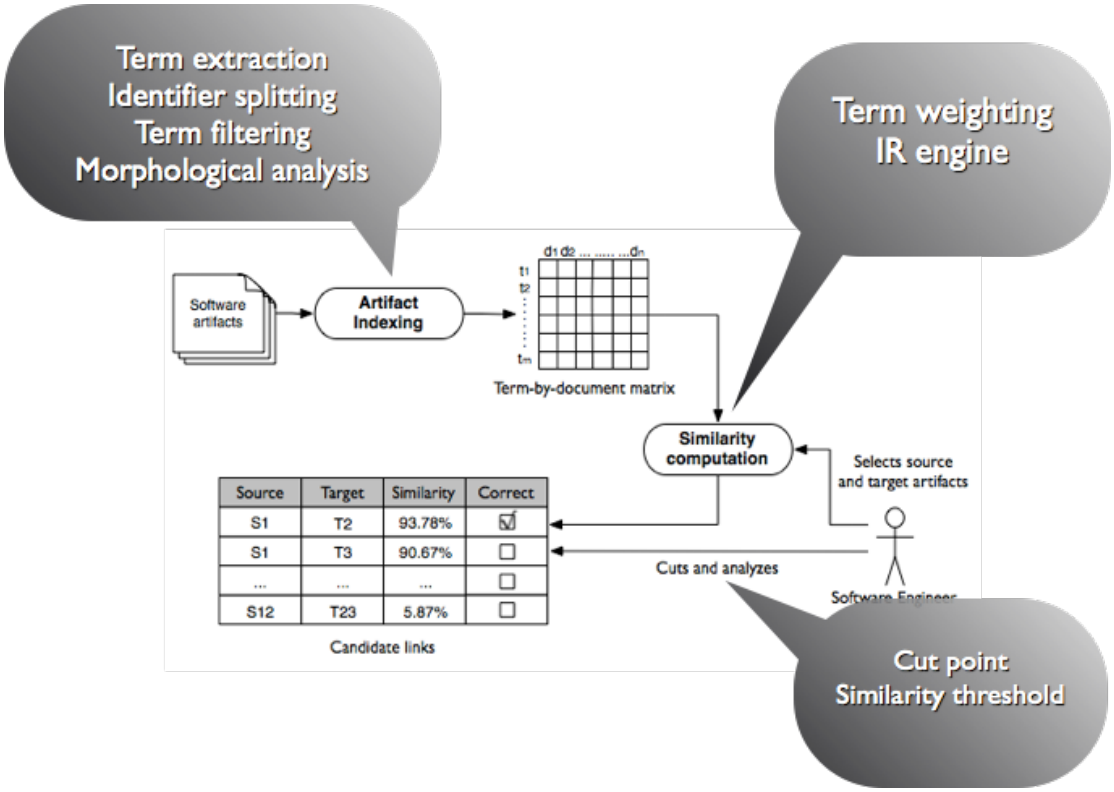

Figure 1. IR-based Traceability recovery: the process.

As Figure 1 shows the output of the IR-based Traceability Recovery process consists in a list of candidates' links. This list contains pairs of software artifacts ordered on the basis of their textual similarity.
For example, given in input a list of software artifacts source "Use Cases" and a list of software artifacts target "Code Classes" (CC1, CC2, CC3…) the output of this process can be as following:

source	target	similarity
UC2	CC1	0.7669920
UC9	CC21	0.7664471
UC1	CC20	0.7604939
UC19	CC32	0.7502923
UC11	CC25	0.7359851
UC7	CC26	0.7185895
UC4	CC18	0.7163126
UC3	CC20	0.6583924
UC4	CC1	0.6488923

……

At this point the software engineer must to classify these proposed links as false positive (the pair of artifacts are not linked) or true positive (the pair of artifacts are really linked) as follow:

source	target	similarity	evaluation
UC2	CC1	0.7669920	1
UC9	CC21	0.7664471	1
UC1	CC20	0.7604939	1
UC19	CC32	0.7502923	1
UC11	CC25	0.7359851	1
UC7	CC26	0.7185895	0
UC4	CC18	0.7163126	1
UC3	CC20	0.6583924	0
UC4	CC1	0.6488923	0

…...

To evaluate the performances of the IR-based Traceability Recovery process an important step is represented by the choice of the ***cut point*** for the similarity threshold (Figure 1). For example if we choose as cut point 0.75 the ranked list above becomes:

source	target	similarity	evaluation
UC2	CC1	0.7669920	1
UC9	CC21	0.7664471	1
UC1	CC20	0.7604939	1
UC19	CC32	0.7502923	1

If we choose as cut point 0.65 the ranked list is as follow:

source	target	similarity	evaluation
UC2	CC1	0.7669920	1
UC9	CC21	0.7664471	1
UC1	CC20	0.7604939	1
UC19	CC32	0.7502923	1
UC11	CC25	0.7359851	1
UC7	CC26	0.7185895	0
UC4	CC18	0.7163126	1
UC3	CC20	0.6583924	0

The evaluation of the accuracy of the output of the IR-based Traceability recovery process is performed using two metrics:

$$precision = \frac{|correct \cap retrieved|}{|retrieved|} \qquad recall = \frac{|correct \cap retrieved|}{|correct|}$$

In particular, the Precision corresponds to the number of true positive in the ranked list divided by the number of proposed (or retrieved) links. Vice versa, the Recall corresponds to the number of true positive in the ranked list divided by total number of correct links. The total number of the correct links corresponds to the entire set of links between the various software artifacts. The Recall is computed in this way to better understand to what extend the list of candidate links contain correct links.

Let us consider the ranked list with cut point 0.75:

source	target	similarity	evaluation
UC2	CC1	0.7669920	1
UC9	CC21	0.7664471	1
UC1	CC20	0.7604939	1
UC19	CC32	0.7502923	1

Let us consider that the total number of correct links is 20. In this example the value precision is equal to 1 (4/4) and the recall is 0.2 (4/20). This means that the proposed approach with a cut point 0.75 the ranked list suggests 20% of total number of traceability links to recover with an accuracy of 100%.

For the ranked list with cut point 0.65 precision and recall are 75% and 30% respectively. It is important to note that with lower threshold the precision tends to decrease while the recall tends to increase.

Exercise:

a) Download the EasyClinic Dataset (a dataset offered by the University of Salerno, Italy) in http://www.ifi.uzh.ch/seal/teaching/courses/softwareWartung-FS16/EasyClinic-Data.zip.
The zip contains three folders:

1. "documents": contains two kinds of software artifacts such as, Use cases and Class Descriptions.
2. "oracle": contains a file named "oracle2.txt" that reports all the correct links (is the traceability matrix prepared by the original developers of the dataset) between the software artifacts.
3. "ranked lists": contains three ranked lists (with different threshold 0.75, 0.65 and 0.55). The ranked list are generate computing the textual similarity (i.e., the cosine similarity) between software artifacts (the file R "script.r" in the zip report the step used to generate all the ranked lists)

b) Execute the script R "script.r".

c) Use the oracle to evaluate the accuracy of the generated ranked list computing the precision and recall (as explained above).

d) Consider the data of the ranked list that has similarity threshold=0.55 (or lower) and plot a graph "Precision - Recall" where the x-axis is represented by the Recall and the y-axis is represented by the Precision. The intervals in the graph should be of 10% for both Precision and Recall. Thus, in this graph we should be able to see the corresponding level of Precision when the Recall is, for example, 10% and vice versa. An example of graph Precision and Recall can be find at the following link: http://www.searchtechnologies.com/images/Precision-Recall-graph.gif

To perform this task and plot the graph you should show the values of precision when the recall in 10%, 20%, 30% etc. Let us for example, to compute the precision when the recall is 10%.

EXAMPLE:

If 120 is the number of total correct links, for us 10% of recall means to recover 12 correct links in the ranked list (120/100*10=12).

Thus, compute the precision when we have recall 10% means to compute the precision when we recover in the ranked list 12 links. This means that the "cut point" to compute the precision is the position in the ranked list in which we recover exactly 12 links.

e) Discuss possible limitations of the IR-based approaches used for Recovery Traceability Links (maximum one page). Discuss possible

alternative way to Recovery Traceability Links among Software Artifacts. References from the literature are very welcome for this task.