


Estimating the number of remaining links in traceability recovery

Davide Falessi¹  · Massimiliano Di Penta² ·
Gerardo Canfora² · Giovanni Cantone³

Published online: 20 October 2016
© Springer Science+Business Media New York 2016

Abstract Although very important in software engineering, establishing traceability links between software artifacts is extremely tedious, error-prone, and it requires significant effort. Even when approaches for automated traceability recovery exist, these provide the requirements analyst with a, usually very long, ranked list of candidate links that needs to be manually inspected. In this paper we introduce an approach called Estimation of the Number of Remaining Links (ENRL) which aims at estimating, via Machine Learning (ML) classifiers, the number of remaining positive links in a ranked list of candidate traceability links produced by a Natural Language Processing techniques-based recovery approach. We have evaluated the accuracy of the ENRL approach by considering several ML classifiers and NLP techniques on three datasets from industry and academia, and concerning traceability links among different kinds of software artifacts including requirements, use cases, design documents, source code, and test cases. Results from our study indicate that: (i) specific estimation models are able to provide accurate estimates of the number of remaining

Communicated by: Patrick Mäder, Rocco Oliveto and Andrian Marcus

✉ Davide Falessi
dfalessi@calpoly.edu

Massimiliano Di Penta
dipenta@unisannio.it

Gerardo Canfora
canfora@unisannio.it

Giovanni Cantone
cantone@uniroma2.it

¹ Department of Computer Science, California Polytechnic State University, San Luis Obispo, CA, USA

² Department of Engineering, University of Sannio, Benevento, BN, Italy

³ Department of Civil Engineering and Computer Science, University of Rome Tor Vergata, DICII, Rome, Italy

positive links; (ii) the estimation accuracy depends on the choice of the NLP technique, and (iii) univariate estimation models outperform multivariate ones.

Keywords Information retrieval · Traceability link recovery · Metrics and measurement

1 Introduction

Traceability links capture relations among software artifacts at the same level —i.e., “vertical traceability” among requirements or use cases—or at different levels i.e., “horizontal traceability”, for example between requirements and source code artifacts (Lindvall and Sandahl 1996). The availability of such links is relevant for several software development tasks. A relevant example of vertical traceability is the identification of equivalent requirements to avoid assigning the same requirement to different requirements analysts, thus performing redundant tasks and originating duplications in software artifacts up to the code base (Dag et al. 2002). This usually happens when requirements are numerous and many stakeholders are involved in the analysis process (Falessi et al. 2011). Traceability links are useful also during maintenance when, given a requirement change, one needs to know what design, test, and source code artifacts need to be changed accordingly.

Due to software evolution, or to the lack of a disciplined development and maintenance process, traceability links often tend to be outdated or missing. Several researchers have proposed the use of Natural Language Processing (NLP) techniques to support traceability recovery among artifacts at the same level of abstraction, e.g., requirements (Falessi et al. 2011), and at different levels of abstraction, including high-level onto low-level requirements (De Lucia et al. 2009; De Lucia et al. 2006), requirements onto design (Duan and Cleland-Huang 2007; Zou et al. 2007; Lormans and van Deursen 2006), requirements onto source code (Marcus and Maletic 2003; Antoniol et al. 2002), requirements onto defect reports (Yadla et al. 2005), and change requests onto the impacted software modules (Antoniol et al. 2000). Such techniques produce a ranked list of candidate traceability links, that need to be manually inspected by the analyst.

Automated support for traceability link recovery is particularly important in case of large-medium systems. However, for such systems, the number of candidate links could be very high; this makes infeasible (with a reasonable effort) a complete manual inspection. For example, in the industrial case study of detecting equivalent requirements discussed by Falessi et al. (2011) there were five projects, each with 500 requirements. Given that a candidate link is a unique pair of requirements, within the same project or across different projects (for datasets composed of more than one project, there can be inter-project traceability links), where the order does not matter, the formula to compute the number of different candidate links is $N(N-1)/2$, where N is the number of requirements. Thus, the number of possible requirement pairs to link were more than three million. In such cases, it is very unlikely that the analyst can inspect all possible links. Thus, an accurate ranking allows the analyst to focus on the links with the highest likelihood to be of interest. However, the ranking is never 100 % precise because it relies on the textual content of the artifacts to be linked. Therefore, the analyst must make the decision to stop searching for artifacts to link by making a trade-off between missing actual links and inspecting a large number of false positives. On the one hand, an overestimation leads to wasting effort, because the search does not produce any valuable output. On the other hand, an underestimation leads to an infective link recovery. For instance, in the context of detecting equivalent requirements, an underestimation would lead to wasting effort in developing duplicate pieces of functionality.

Paper Contribution To support the requirements analyst in the decision about when to stop inspecting candidate traceability links, in this paper we introduce a novel approach called Estimating the Number of Remaining Links (ENRL) which aims at providing the analyst with an estimate of how many positive links the remaining part of the ranked list contains. ENRL consists of a Machine Learning (ML) classifier that is trained on the portion of the ranked list already inspected by the analyst and uses the similarity between the two items to be linked (measured by an NLP technique) as feature (i.e., input of the ML classifier). We note that the number of remaining links differs from the metric recall because it relates to a ranked set of classified instances.

To empirically evaluate the accuracy of ENRL, we performed a study on three datasets with seven ML classifiers and several (12 for each dataset) NLP techniques. The ML classifiers are used as univariate models (i.e., using as feature the artifact similarity computed by one single NLP technique) and multivariate ones (i.e., using as features the artifact similarity computed by all the considered NLP techniques). One dataset regards five industrial projects and the linkage of equivalent requirements, the other two datasets are academic and concern traceability recovery among different kinds of software artifacts such as use cases, design documents, source code, and test cases.

The purpose of this work is exploratory: we aim to analyze if, and how, we can estimate the number of remaining links in traceability recovery. In this context, the study aims at investigating the following research questions:

- **RQ1:** What is the level of accuracy of estimation models?
- **RQ2:** Can we select accurate estimation models?
- **RQ3:** Do multivariate estimation models outperform univariate ones?

Paper Structure The remainder of the paper is structured as follows. Section 2 describes the NLP-based link recovery process, and discusses related work. Section 3 describes the ENRL by detailing the role of NLP techniques and ML classifiers. Section 4 reports the empirical study definition and planning. Results are reported in Section 5, while Section 6 discusses the threats to results validity. Section 7 concludes the paper and outlines directions for future work.

2 Related Work

This section describes related work for what concerns (i) traceability link recovery, and (ii) the use of various statistical techniques to estimate the number of items (e.g., of defects) that are remaining or relevant for a specific task (e.g., a testing task). We forward to a recent systematic literature review (Borg et al. 2014) for a more broad discussion of the state of the art on traceability link recovery.

2.1 Information Retrieval Approaches for Traceability Link Recovery

As depicted in Fig. 1, Information Retrieval (IR) traceability link recovery works by textually comparing pairs of software artifacts, and ranking the similarity of artifact pairs (Falessi et al. 2011). NLP techniques have been used for a long time to recover traceability links (Borg et al. 2014). In practice, NLP techniques analyze the two textual artifacts and provide a similarity measure ranging between 0 and 1. This similarity measure represents the likelihood of the two textual artifacts to be related. Generally, the artifacts pairs are ranked

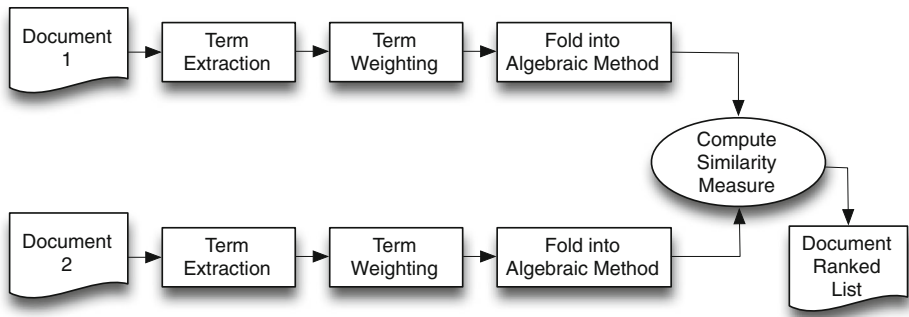


Fig. 1 IR-based traceability recovery

according to their similarity scores and the ones having a similarity higher than a given threshold (therefore being in the topmost positions of the ranked list) are considered as candidate links. An automated recovery is often followed by a manual inspection of the candidate links; it is performed by a software analyst who can confirm a candidate link, or classify the link as a false positive.

NLP techniques consist of a sequence of steps, described as follows (further details can be found in well-known textbooks (Baeza-Yates and Ribeiro-Neto 1999)):

- *Term extraction*: terms are extracted from software artifacts, removing special characters, and possibly splitting composite identifiers before any comparison can take place. Possible ways to extract terms include: 1) Simple extraction (tokenization and stop-word removal) and 2) Part Of Speech (POS) Tagging using the Stanford natural language parser, which would allow to focus on some particular POS only, such as nouns (Capobianco et al. 2009), and 3) Stemming, i.e., bringing different word forms (noun singular/plural, verb conjugations) to the same radix, using approaches such as the Porter stemmer (Porter 1980; Fellbaum 1998);
- *Term weighting*: terms are weighted, based on their occurrences in the analyzed text fragments. For instance the weight of a term can depend on the frequency of its occurrence in the given text fragments (term frequency–*tf*) or the number of documents that include the term (Inverse Document Frequency–*idf*), or their combination (*tf-idf*) (Baeza-Yates and Ribeiro-Neto 1999). Simpler approaches include the Raw frequency, i.e., the number of times that the term occurs in the text, and Binary weighting, where a weight of 0 is assigned if the term does not occur, and 1 vice versa.
- *Building algebraic models*: a space of documents is created. In the case of Vector Space Model (VSM), it is a term-document matrix obtained using the indexing computed at the previous step. VSM uses spatial proximity for semantic proximity. Two further techniques that consider synonymies are WordNet, where synonymies are pre-defined in a thesaurus, and Latent Semantic Analysis (LSA) (Deerwester et al. 1990; Blei et al. 2003), which projects the documents into a (reduced) space of orthonormal concepts to deal with the limitations of VSM such as synonymy and homonymy.
- *Computing similarity*: finally, the similarity between documents to be linked is computed using similarity measures. Specific algebraic models are compliant to specific measures. Examples of similarity measures are Cosine, Jaccard, and Dice similarity

metric for the VSM algebraic model (Baeza-Yates and Ribeiro-Neto 1999). Relevant measures for the WordNet algebraic model include Resnik, Lin, Jiang, and Pirró and Seco (Baeza-Yates and Ribeiro-Neto 1999).

Different authors used different NLP techniques to recover traceability links. For example, Antoniol et al. (2002) used the simple VSM as well as a probabilistic model. Marcus and Maletic (2003) replicated the study of Antoniol et al. (2002) and showed that LSA (Deerwester et al. 1990) in some cases improves the performance of link recovery, thanks to its capability of dealing with synonymy and homonymy. Abadi et al. (2008) applied the Jensen and Shannon Model (JS) (Cover and Thomas 1991), which, again, in some cases outperforms simpler techniques. Other examples of NLP techniques applied to link recovery include Latent Dirichlet Allocation (LDA) (Blei et al. 2003), used by Asuncion et al. (2010), or Relational Topic Models (RTM) used by Gethers et al. (2011). The study by Gethers et al. (2011) indicated that, in general, different techniques perform better or worse in different contexts, and there is no technique able to always outperform the others. In a previous paper (Falessi et al. 2011) we observed that the performances of an NLP technique in identifying equivalent requirements, computed on given datasets, depend on both its ability and the odds of making correct identifications. In order to avoid this problem we proposed, and applied on 242 NLP techniques, the following empirical principles: 1) Definition of an objective function and adoption of an optimal threshold, 2) Cross-validation of the performance of NLP techniques, 3) Statistically testing the difference among performances of NLP techniques, 4) Statistically testing the equivalence among NLP Techniques, 5) Analyzing the impact of difficulty, 6) Analyzing boundaries by adjusting scores, and 7) Validation of the random score.

Recent work has also shown that near optimal IR processes can be instantiated by using search-based optimization techniques (Lohar et al. 2013; Panichella et al. 2013). The existing literature also reports works aimed at improving the performances of traceability link recovery approaches. Settini et al. (2004) proposed to normalize document lengths using the pivot normalization term weighting approach. The purpose is to avoid that longer documents tend to achieve a higher score than shorter ones. Other approaches aim at improving the term weighting by taking into account the structure of the artifacts (Cleland-Huang et al. 2005) or the importance of a term for a specific domain (Zou et al. 2007; 2010; Huffman Hayes et al. 2003). De Lucia et al. (2011) proposed an approach, based on image filtering, to improve the performances of NLP-based link recovery, by weighting words according to specific types of artifact (e.g., use cases, source code) to be linked. Results indicate that the use of smoothing filters significantly improves the performances of IR-based traceability recovery based on VSM or LSI. Feedback mechanisms, such as the Rocchio feedback (Baeza-Yates and Ribeiro-Neto 1999), allow to change the term weighting during the analysis of candidate links performed by software engineers (De Lucia et al. 2006; Hayes et al. 2006). If the software engineer classifies a candidate link as a correct link, the words found in the target artifact increase their weights in the query, otherwise, such weights decrease. The expectation is retrieving more correct links and less false positives in next iterations of the recovery process. All the above techniques aim at improving the performance of link recovery. However, while such techniques generally work toward achieving a better precision/recall, or by trying to move positive links towards the upper part of the ranked list to be presented to the analyst, they have never been used to provide the analyst with information about when to stop searching for artifacts to link. Borg et al. (2014) recently provided a comprehensive analysis of 79 publications related to IR-based trace recovery. By providing

our tool as open source and our measurement procedure as pseudo code we followed their suggestion to report tool details and measurement procedures.

Our work shares with previous traceability recovery approaches the use of NLP techniques to compute the similarity between artifacts to be traced. However, our work adds on top of that the machine learning approach used to estimate the number of remaining links.

2.2 Industrial Applications of Automated Traceability Link Recovery

As described above, researchers have put substantial effort in developing approaches for traceability link recovery, as well as possible ways to improve them. The general goal has been to achieve performances suitable to industrial applicability.

In recent and past years, traceability link recovery has also been applied in the context of industrial case studies. For example, Lormans et al. have developed an approach named *ReqAnalyst* and based on LSI to reconstruct requirement views (Lormans et al. 2008), and successfully applied it to an industrial case study. Noticeable applications in the industrial context of traceability link recovery regard both tracing architectural concerns in safety-critical and performance-critical applications (Mirakhorli and Cleland-Huang 2011) and providing just-in-time traces for large and complex mechatronic systems (Czuderna et al. 2012). Approaches have also been developed to trace quality concerns in business and safety-critical systems (Mirakhorli et al. 2012). Last, but not least, traceability has been combined with design slicing to help filtering out irrelevant content in System Modeling Language (SysML) design when performing inspections (Briand et al. 2014).

2.3 Approaches for Estimating the Number of Items Relevant for a Specific Task

Related literature also concerns approaches to estimate the number of items with specific characteristics in a set, e.g., the number of fault-prone classes in a system. To the best of our knowledge, no past work aimed at investigating the use of ML classifiers to estimate the number of remaining positives in a ranked list. A well-known type of models for estimating the number of remaining items is “capture-recapture”. This type of estimation model has a widespread use in biology (Otis et al. 1978) and its accuracy has been investigated also in software engineering in the context of estimating remaining defects in software artifacts (Briand et al. 2000; Petersson et al. 2004). This type of estimation model uses several persons to inspect the code. Each person draws a sample from the population of defects in the inspected software artifact. A defect discovered by one person and rediscovered by another is said to be recaptured. The percentage of recaptured defects, over non recaptured defects, is a good indicator of remaining defects. Specifically, the higher the percentage of recaptured defects the lower the number of remaining defects. The main advantage of capture-recapture models is their independence from the type of item for which the remaining number is being estimated; these models can be used to estimate the number of remaining penguins in Antarctica or the number of remaining defects in a software application. However, all capture-recapture models require that a high number of analysts search for the desired item (e.g., equivalent requirements or software defects) in an independent way. This assumption is unrealistic in the context of identifying requirement pairs where the analysts must follow a ranked list of candidate requirement pairs rather than advancing subjectively among the wide spectrum of candidates (Cuddeback et al. 2011; Dekhtyar et al. 2011). In conclusion, though addressing a very similar problem, the capture-recapture type of estimation model is incompatible to our context, and therefore we choose to investigate the use of ML classifiers.

A number of studies investigated the estimation of residual defects with the aim to provide further insights into the software testing process. Malaiya and Denton (1998) proposed an approach based on test coverage to estimate the number of residual defects in software applications. Their approach refines the standard use of the exponential software reliability growth model (SRGM) which estimates the total number of defects present at the beginning of testing less the ones found during testing, by taking into account the applied testing effort and specific testing techniques effectiveness. Bai et al. (2008) provide a general model to study the trend of the remaining software defect estimation (RSDE), whereas an overview of different approaches for that has been provided by Cai (1998). Approaches based on reliability models or on natural models such as capture-recapture are not suitable to be applied to the estimation of traceability links, as they are based on the use of defect-specific curves. The most related models are, instead, Bayesian models that have been applied for the estimation of the number of remaining defects. In our case, ML approaches—including Bayesian models—are combined with the use of NLP techniques.

In the context of behavioral testing of software applications, Chen et al. (1999) presented and evaluated an approach to set a stopping rule for determining when to switch testing strategies for increased code coverage or stop testing behavioral models all together. Although setting a stopping rule is similar to estimate the number of remaining defects, their context of behavioral models differs from the context of traceability links retrieval.

3 Estimating the Number of Remaining Links

Figure 2 shows how an NLP technique supports a requirements analyst in identifying positive links, and the ENRL approach, i.e., how ML-based estimator helps in estimating the number of remaining (positive) links.

There are three main steps in the identification of candidate links using IR-based traceability link recovery. In *step 1*, the artifacts are extracted from a repository (e.g., a requirements management tool, an SVN repository, etc.) and each pair of artifacts becomes a candidate link. In *step 2*, an NLP technique analyzes the possible links and computes the similarity for each artifacts pair. Then, artifact pairs are ranked according to the similarity score. In *step 3*, the analyst, following the ranked list, inspects two software artifacts and s/he classifies the link as either positive or negative.

ENRL enhances the standard traceability process with three additional steps, as shown in Fig. 2. Specifically, in *step 4* a ML classifier is trained with the set of already analyzed and classified (into positive or negative) links. The estimation model is trained using the similarity scores provided by an NLP technique (in *step 2*), and the specific classification provided by the human (in *step 3*). In *step 5*, the trained model takes as input the similarity scores of the remaining (not yet analyzed) links, and classifies them as positive or negative links. In *step 6* the estimated number of remaining links is computed as the number of links estimated as positive by the ML classifier. Such an estimate is an objective information that can be used by the analyst—as an alternative or a complement to her subjective intuition—to decide whether it is worthwhile to proceed in the inspection of candidate traceability links or not.

It is important to highlight why the number of remaining links is estimated by using a ML classifier, rather than just setting a threshold on the similarity score and counting the number of candidate links whose similarity is above the threshold. The main reason is that an NLP technique performs differently on different thresholds: a high threshold is prone to false negatives and therefore an underestimation of the number of remaining links, whereas

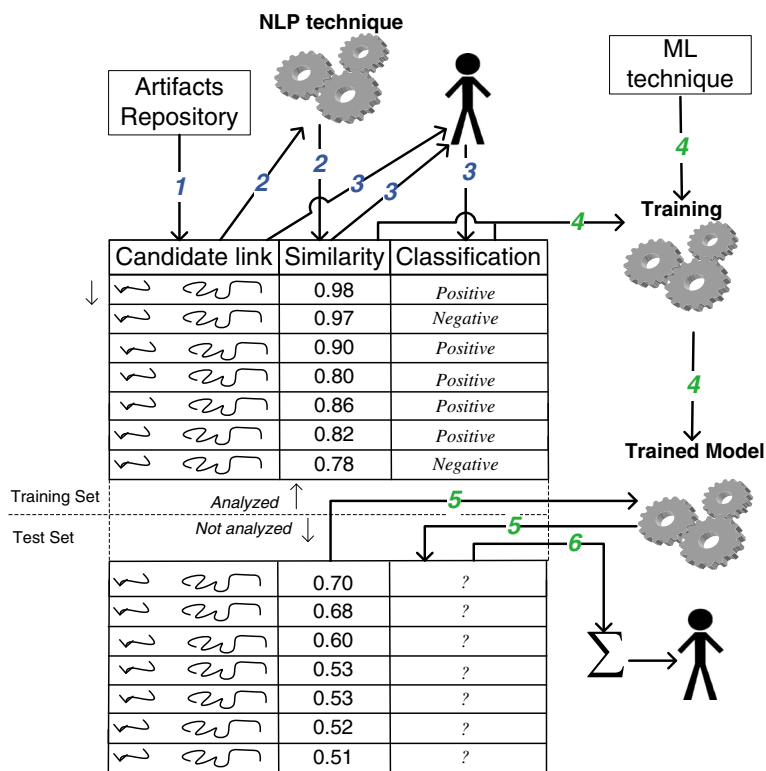


Fig. 2 The ENRL approach: estimating the number of remaining links using NLP techniques and ML classifiers

a low threshold is prone to false positives and therefore produces an overestimation of the number of remaining of links (De Lucia et al. 2007). As a result, for each NLP technique, a given threshold may or may not be suitable. Each ML classifier can be seen as a way to tune the threshold according to a specific algorithm. This justifies the use of a supervised technique (i.e., using ML classifiers) over an unsupervised one (i.e., setting a threshold over the computed similarity levels).

4 Empirical Study

The *goal* of this study is to analyze ENRL approach, with the *purpose* of investigating the capability of ML classifiers to estimate the number of remaining links in a ranked list produced by NLP link recovery approaches. The *quality focus* is the ability of ML classifiers to support ENRL. The *perspective* is of researchers that want to investigate the ENRL accuracy, and whether its accuracy can be considered as an additional quality factor, i.e., a criteria to be used when selecting NLP approaches for traceability link recovery. The *context* comprises candidate traceability links from three datasets, including one industrial dataset and two datasets, related to student projects, widely used in traceability recovery studies.

Table 1 Characteristics of the three datasets

Project	Artifacts			Candidate	Positive
	Kind	Number	Total	Links	Links
Selex SI	Requirements	2,500	2,500	983	138
ETour	Use cases	58	174	6,768	385
	Code classes	116			
EasyClinic	Use cases	30	160	25,440	1,618
	Interaction diagrams	20			
	Source code classes	63			
	Test cases	47			

4.1 Context Selection

As detailed in Table 1, the first dataset (Selex SI) comes from industry—specifically from Selex Sistemi Integrati¹—and was used in previous studies (Falessi et al. 2011; Falessi et al. 2009). It consists of 983 candidate traceability links where each link is a requirement pair. This dataset features 138 true positive links sampled from all possible 3,123,750 pairs of 2,500 requirements. Such requirements belong to a system of systems composed of five industrial projects from the aerospace domain. Noteworthy, and differently from the other datasets, in this case there can also be inter-project traceability links. Further details about this dataset and the adopted sampling procedure can be found in a previous paper (Falessi et al. 2009). The second and third datasets are related to two Java projects, *e-Tour* and *EasyClinic*, developed by Master Students and related to e-tourism and hospital management respectively. For *e-Tour*, the traceability link recovery is performed between use cases and source code classes, whereas for *EasyClinic* traceability recovery is related to four kinds of artifacts: use cases, textual information from interaction diagrams, source code classes, and test cases. In *EasyClinic*, traceability link recovery concerns both horizontal and vertical traceability (e.g., use cases vs. source code classes). We complemented the industrial dataset with *e-Tour* and *EasyClinic* because (i) they allow to show the approach applicability to vertical traceability recovery, and (ii) these data sets are publicly available and have been previously used in a traceability recovery challenge.² The working dataset is available online³ for replication purposes.

4.2 Research Questions

In this paper we aim to investigate the following research questions:

- **RQ1:** *What is the level of accuracy of estimation models?* This research question aims at understanding the accuracy of available ML classifiers in supporting ENRL. Moreover, since several ML classifiers and NLP techniques are available, we investigate if the accuracy depends on combination of the specific ML classifier and NLP technique adopted. For example, different ready-to use tools can adopt specific techniques, e.g., may use LSI instead or VSM or vice versa. Furthermore, the quality of an NLP may

¹<http://www.finmeccanica.com/en/home>

²<http://www.cs.wm.edu/semeru/tefse2011/Challenge.htm>

³<http://www.ing.unisannio.it/mdipenta/estimating-links.tgz>

- vary among contexts, e.g., one can decide to adopt VSM when the corpus is fairly limited, or LSI for larger corpora that allows the learning. Thus, we are interested in investigating: i) which combination of ML classifier and NLP technique provides the highest accuracy and ii) the level of this accuracy.
- **RQ2:** *Can we select accurate estimation models?* In order to provide practical benefits, estimation models must not only be sufficiently accurate but must also be selectable (i.e., we need to know that it will be accurate for the case at hand). In other words, having an accurate estimation model would be useless if the final user would be unable to select it among inaccurate models. Unfortunately, the accuracy of an estimation model can be measured only after its use and thus this measure cannot be used as criteria for selecting estimation models. In this work we investigate whether the accuracy in ENRL can be estimated. If this will be the case, then the estimated accuracy can be used as a reliable criteria for selecting estimation models. Moreover, because in this study an estimation model is a specific combination of an ML classifier and an NLP technique, in **RQ2** we investigate whether, for a given ML classifier, it is possible to choose the NLP technique according to the estimated accuracy. It is important to point out that, while the performances of the proposed approach might depend on the particular choice of the ML technique, our goal is to show that the approach for predicting the estimated number of remaining links could produce acceptable performances with out-of-the-box techniques, and without necessarily performing a careful selection and calibration of such techniques.
 - **RQ3:** *Do multivariate estimation models outperform univariate ones?* Previous research indicates that combining several NLP techniques yields improvements in accuracy, precision, and recall (Falessi et al. 2011). Thus, we are interested in investigating if there is an improvement in estimation accuracy too. In other words, we investigate if ML classifiers adopting several NLP techniques simultaneously (i.e., multivariate models) provide more accurate estimations than ML classifiers adopting one NLP technique at a time (i.e., univariate models).

4.3 Variable Selection: Independent Variables

The only independent variable of this study is the estimation model adopted to support ENRL. Since in our context an estimation model is a specific combination of ML techniques and NLP techniques, in the following we describe these two variables.

4.3.1 Machine Learning Supervised Classifiers

To support ENRL we considered the ML classifiers available in Weka. We chose WEKA among several tools implementing ML classifiers (e.g., Mallet, NLTK, R, MATLAB) due to the availability of training support (Witten et al. 2011), the extensive use in the scientific research community (Kim et al. 2011; Okutan and Yildiz 2014; Rahman et al. 2013; Krishnan et al. 2013), its open-source nature, and the first author acquaintance (Falessi et al. 2011; Falessi and Reichel 2015; Falessi et al. 2014). On top of Weka we developed a tool called ATHLETE (Automated macHine LEarning Techniques Evaluation). ATHLETE is distributed under GPL v2 license and publicly available.⁴ ATHLETE applies the measurement procedure required in this study by adopting the ML classifiers available via the

⁴<https://github.com/apicchiani/estimationTool>

Weka API. In the absence of any previous study on estimating the number of remaining links, and given the large spectrum of ML classifier types available, we have randomly chosen 7 ML classifiers with the constraint to have one classifier for each Weka category, namely Bayesian, Functions, Lazy, Meta, Misc, Rules, and Trees. Specifically, the chosen ML classifiers are:

1. *ADTree*, which is a boosting technique employing a decision tree (Freund and Mason 1999);
2. *Bagging*, which uses bootstrapping techniques to train classifiers on the training set (Breiman and Breiman 1996);
3. *FLR (Fuzzy Lattice Reasoning)* classifier (Athanasiadis 2007);
4. *IBk*, which uses K-nearest neighbor clustering for performing a classification (Altman 1992);
5. *NaiveBayes*, which is a probabilistic classifier based on the Bayes conditioned probability theorem (Russell and Norvig 2003);
6. *LogitBoost*, which is a classifiers using a boosting algorithm (Friedman et al. 2000);
7. *ZeroR*, which is a constant classifier, which always returns as classification the most one, and is used as a baseline when assessing the performances of other classifiers.

In this context it is not necessary to understand the details of the various classifiers, because the goal of this paper is not to try building the best possible classifier to support ENRL. Instead, we put ourselves in practitioners' shoes and we investigate whether out-of-the-box classifiers (e.g., available in tools such as Weka) can be used to support ENRL, and to investigate to what extent the choice of one particular classifier would affect the error in the estimate.

4.3.2 NLP Techniques

When choosing the set of NLP techniques to be used, in principle one could have all possible combinations of different methods available for each phase of the traceability recovery process described in Section 2. In Falessi et al. (2011) we described about 242 NLP techniques. Each technique can be seen as a specific point in a four-dimensional space where each dimension is a mechanism (i.e., algebraic models, term extraction, weighting schema, and similarity metric). We choose the 12 NLP techniques with the aim to use all the alternatives of all dimensions, though not all their possible combinations because this would lead to 242 NLP techniques. For instance, regarding Selex SI, we used the “Jaccard” similarity metric alternative in combination with the “VSM” algebraic model alternative, and the “Cosine” similarity metric alternative in combination with the “VSM” algebraic model alternative. We believe that 12 is a high number of techniques because the goal of this paper is not to investigate which technique is better in terms of traceability recovery capabilities; see Falessi et al. (2011) for a comparison of 242 NLP techniques in terms of precision, recall, and accuracy in traceability recovery. It is important to note that LSA required to set the number of concepts k to different values in different projects, given the different size of the datasets.

4.4 Variable Selection: Dependent Variables

The main dependent variable of this work is the accuracy of estimation models. In order to comprehensively characterize the various aspects of accuracy of an estimation model, we will use several metrics.

4.4.1 Mean Relative Error (MRE)

A widely adopted metric to assess the accuracy of estimation models is the relative error (Foss et al. 2003) which in our case is the absolute value of (*actual number of remaining positive links* - *estimated number of remaining positive links*) / *actual number of remaining positive links*. Given the set of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN), the relative error (RE) is computed as:

$$RE = \frac{|(TP + FN) - (TP + FP)|}{TP + FN} = \frac{|FN - FP|}{TP + FN} \quad (1)$$

Note that we have $RE = 0$ when $FP = FN$, regardless of the size of FP (and FN); though surprising, this is fine. In fact, given a set of remaining links, we aim to estimate the number of positive links not which of them are positive. Specifically, if the number of remaining links is X , there might be a very rare case in which an approach A_1 is able to identify most of these positive links and another approach A_2 is unable to identify these positive links, while able to estimate their number X . This can indeed happen because, when estimating the number of links, the presence at the same time of a false positive and a false negative (which count as a total number of links equal to one) produces the same results when the false link is discarded and the true link identified (number of links equal to one also in this case).

In this scenario, we definitely prefer the approach A_2 over A_1 as, differently than past research on traceability recovery, the problem we want to solve is the quantification, rather than the identification, of positive links. We do believe this case is rare and probably only hypothetical.

The MRE is defined as the mean RE across all performed estimations in a group of datasets, i.e.:

$$MRE = \frac{\sum_{i=1}^N RE_i}{N} \quad (2)$$

where N is the number of performed estimations.

4.4.2 Mean Absolute Error (MAE)

The absolute error (AE) complements the RE as it provides a clear, intuitive indication of the difference between the estimated links and the actual ones. AE is defined as:

$$AE = |(TP + FN) - (TP + FP)| = |FN - FP| \quad (3)$$

The Mean Absolute Error (MAE) is defined as the mean AE across all performed estimations in a group of datasets, i.e.:

$$MAE = \frac{\sum_{i=1}^N AE_i}{N} \quad (4)$$

where N is the number of performed estimations.

Despite MRE and MAE result with the same trend on the same dataset, they have different values and different meanings (i.e., relative versus absolute error). Therefore, both MRE and MAE are required to support a reliable interpretation of the results.

4.4.3 Standard Deviation of RE (SD)

Standard deviation aims at complementing the use of MRE as the latter does not take into account the spread of the RE distribution across the performed estimations. In other words,

an estimation model, to be deemed as accurate, it has to show a little variance of accuracy among estimations. Note that this metric cannot be used alone as it can erroneously prefer a ML classifier, which constantly performs erroneously.

4.4.4 90 Quantile of RE (90Q)

A further important aspect in the definition of accuracy of an estimation model is its probability to provide estimations with low MRE. By using quantiles, we set the minimum proportion of observations exhibiting a given maximum relative error.

4.5 Study Design and Procedure

In the following, we describe in details how we address the study research questions.

4.5.1 RQ1: What is the Level of Accuracy of Estimation Models?

Cross-validation (Stone 1974) is one of the most used and reliable procedure for measuring the accuracy of estimation models (Witten et al. 2011). In a ten-fold cross-validation, the dataset is randomly split into 10 sub-parts, with the constraints, for each part, to have approximately the same cardinality and the same proportion of categories of the data set. Each sub-part, in turn, is taken away from the data set, and the remaining data are used to train the model which is then applied on the subtracted part. However, ten-fold cross-validation is meant to be applied in context where the estimation is independent of ordering of data (candidate links) constituting the training and the test set. Since in our context the ordering matters, ten-fold cross-validation is not suitable. In our context, the aim of estimates is to provide information to analysts, enabling them to make informed decisions about when to stop searching for artifacts to link. Therefore, the smaller the size of training set required to estimation models for providing accurate estimates, the sooner the analyst can take advantage of the information regarding the estimated number of remaining links. Therefore, we designed our cross-validation aiming at mimicking a realistic scenario where—as shown in Fig. 2—the analyst starts to inspect links in the top part of the ranked list. In this scenario, only the top links can be used for training and the remaining (candidate) links are estimated.

We analyze the performances of each estimation model adopting 91 subdatasets for each dataset. In each subdatasets the training part has been constructed by starting with 5 % of the dataset and improving using a step of 1 % of the dataset, up to reaching a subdataset having a training set of 95 % of the data set. We started creating the subdatasets at 5 % rather than at 0 % for providing to the estimation models enough data for learning. We stopped at 95 % rather than 100 % because the approach is meant to suggest when to stop classifying and hence it would be useless if the entire dataset would be already classified. Even reaching a 95 % of classifications could be practical infeasible, however the related results have scientific significance. Finally we adopted a step of 1 % to obtain fine-grained results.

Algorithm 1 reports our procedure for computing the estimation error in pseudocode. The first phase (lines 2 to 8) is to set variables like datasets, ML classifiers, NLP techniques, and the constraints about the subdatasets size. The second phase is the evaluation itself. We select a given dataset and a given NLP technique. Then we rank (line 13) this dataset according to this NLP technique. Once ranked, the dataset is split in training set (line 15) and test set (line 16). The training set includes always the top links, its size varies (line 14) according

to predefined variables (lines 6 to 8). Then we create an estimation model by selecting a ML classifier and training it on the training set (line 18). Finally, the error of a specific dataset, NLP technique, amount of data, and ML classifier is now computed by applying the estimation model on the test set. The procedure continues by changing the ML classifier, the amount of data, the NLP technique and the dataset, in this specific order.

Algorithm 1 Pseudo code of the procedure for computing the estimation error

```

1 begin
2   Datasets[] ← {SelexSI, ETour, EasyClinic}
3   MLClassifiers[] ← {ADTree, Bagging, FLR, IBk, Logit Boost, Naive Bayes}
4   NLPTechniques[] ← {Porter stemmer-IDF-VSM-JACCARD, Porter
   stemmer-RAW-VSM-DICE, Simple-IDF-WordNet-PIRROSECO, Simple-TFIDF-LSA
   30-COSINE, Stanford nouns and verbs-IDF-VSM-DICE, Stanford nouns and
   verbs-IDF-WordNet-JIANG, Stanford nouns-BINARY-LSA 10-COSINE, Stanford
   nouns-IDF-WordNet-RESNIK, Stanford nouns-TFIDF-VSM-COSINE, Stanford
   nouns-TFIDF-LSA 100-COSINE}
5   MinimumTrainingSet ← 5%
6   MaximumTrainingSet ← 95%
7   Step ← 1%
8   foreach dataset ∈ DataSets do // Varying the Dataset
9
10    foreach nt ∈ NLPTechniques do // Varying the NLPTechnique
11
12     // The NLP technique is applied to compute the similarity score among all
       combinations of artifacts in the dataset
13     dataset.ComputeSimilarityScore(nt)
14     // The dataset is now ranked according to the similarity score
       dataset.Rank()
15     for z=MinimumTrainingSet to MaximumTrainingSet stepBy Step do
16       // Varying the amount of data used as training or test
17
18       // The training set is now created by taking the top z% instances from
       the dataset
19       TrainingSet ← dataset.Top(z)
20       // The test set is now created by taking the last (100 - z)% instances
       from the dataset
21       TestSet ← Datasets[i].Last(100-z)
       foreach mlc ∈ MLClassifiers do // Varying the ML classifier
22
23         // estimation model is created by training the ML classifier on the
       training set
24         estimationModel.Train(mlc, TrainingSet)
25         // The error of a specific dataset, NLP technique, amount of data,
       and ML classifier is now computed by applying the estimation
       model on the test set
26         Error[dataset,nt,z,mlc] ← estimationModel.Test(TestSet)

```

To check whether an estimation model is accurate, we set a threshold on the performances—in terms of our dependent variables—observed for each estimation model; for instance we might require $MRE < 0.5$. We started analyzing the thresholds adopted in other software engineering domains (Foss et al. 2003); for instance, in the context of effort estimation, the usual threshold for MRE to deem an estimation model as accurate is 0.25 (Foss et al. 2003). Then, by discussing with our industrial partners (i.e., Selex SI), we adjusted such thresholds. As a result, in this work we deem an estimation model as accurate if it satisfies all the following thresholds: $MRE < 0.15$, Standard deviation (SD) of RE < 0.15 , 90 quantile (90Q) of MRE < 0.30 (i.e., we guarantee that across all performed estimations, more than 90 % of them show a RE smaller than 0.30) and $MAE < 0.05$.

number of positive links in the specific dataset. This set of thresholds takes into account both the average amount of error (MRE and MAE) and its variance (SD and 90Q). However, thresholds vary among contexts as different situations require support of different levels of precision; e.g., in general, safety critical contexts require higher levels of precision than non critical contexts. In order to improve the applicability of the results on other contexts, we analyze the performance of the estimation models, over the three datasets, by applying various thresholds, and specifically MRE threshold varying between 0.05 and 1 with a step of 0.05, and the thresholds on other dependent variables (MAE, SD, 90Q) varying proportionally to the MRE threshold.

4.5.2 RQ2: Can we Select Accurate Estimation Models?

Again, cross-validation is one of the most used and reliable procedure for measuring the accuracy of estimation models. We applied cross-validation to investigate RQ1, however we need to be careful in which data is known, and hence can be used, when the validation is applied. According to Fig. 2, the analysts, while analyzing candidate links, has only a portion of the dataset that is classified and hence only this portion can be used to estimate the accuracy that the estimation model will have in estimating remaining (i.e., unclassified) links.

We define *actual* MRE as the MRE measured via the evaluation procedure reported in Algorithm 1. We define *estimated* MRE as the MRE computed via ten-fold cross-validation (Stone 1974) having as dataset only classified links. According to Algorithm 1, the estimated error is computed by adding the following line after line 19:

EstimatedError = *estimationModel.TenfoldCrossValidation(MLClassifiers[k], TrainingSet)*

Therefore, the training set of actual MRE (Fig. 2) is used as both training and test set when measuring the estimated MRE; the test set of actual MRE is not used (because unknown in the practical scenario) when measuring the estimated MRE.

We analyze the Spearman's ρ and Kendall's τ correlation between the actual MRE and the estimated MRE. A strong correlation would suggest that choosing the estimation model according to the estimated accuracy will likely lead to use the estimation model with the actual high accuracy.

4.5.3 RQ3: Do Multivariate Estimation Models Outperform Univariate Ones?

For each of the seven ML classifiers, we compare the estimation accuracy (i.e., actual MRE) when ML classifiers adopt the similarity scores provided by all NLP techniques (i.e., multivariate) versus the similarity scores provided by one NLP technique (i.e., univariate). Thus, in the multivariate models, the NLP techniques have been combined by simply using them as different features of the ML classifiers. Regarding univariate models, we note that the NLP technique can be chosen in two different ways: the actual or the expected best NLP technique (see RQ2).

5 Study Results

This section reports the study results, addressing the research questions formulated in Section 4.2.

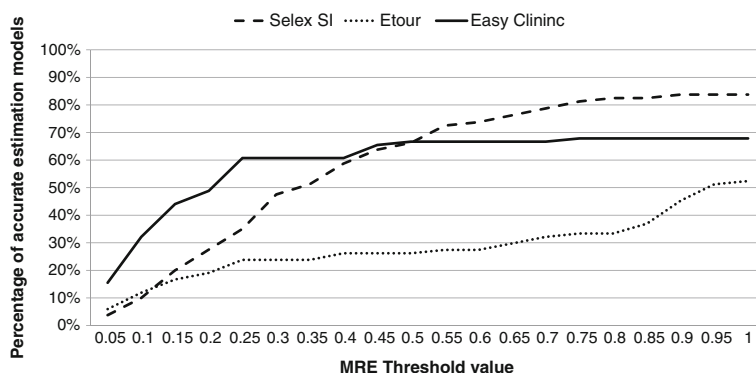


Fig. 3 Relation between MRE threshold and percentage of accurate models obtained

5.1 RQ1: What is the Level of Accuracy of Estimation Models?

Figure 3 reports the percentage of estimation models (y-axis) that are accurate according to the threshold defined in the x-axis. The x-axis reports only the value of the MRE threshold, whereas the thresholds on SD, Q90, and MAE are increased proportionally to the MRE threshold, as explained in Section 4.5. From Fig. 3 we notice that the percentage of accurate models changes, among datasets, according to the specific adopted thresholds. For instance, if we consider the minimum threshold (i.e., $MRE = 0.05$), the dataset having the larger proportion of accurate estimation models is EasyClinic, while Selex SI exhibits the lowest proportion. However, when considering higher thresholds (i.e., $MRE > 0.5$), the dataset having the larger proportion of accurate estimation models is Selex SI, while ETour exhibits the lowest proportion.

Table 2 reports the accuracy—in terms of minimum MRE achieved over the different sizes of the training set—of different ML classifiers. Each row identifies a specific NLP technique,⁵ whereas each column identifies a specific ML classifier. Table 2 reports in boldface the estimation models that are accurate, i.e., the ones satisfying the following thresholds: $MRE < 0.15$, $SD < 0.15$, $90Q < 0.30$, and $MAE < 0.05$. Table 2 highlights in gray the “best” estimation model for each dataset, i.e., the one satisfying the lowest thresholds on MRE, MAE, STDV, and Q90. We report online⁶ additional results about the accuracy of the different ML classifiers.

As Table 2 shows, several combinations of NLP technique and ML classifier are accurate. Therefore, the ENRL approach resulted accurate, although the estimation accuracy varies among the different datasets. In summary, we notice that:

- as expected, the *ZeroR* classifier (not reported in Table 2) never passes the thresholds and always achieves a MRE of 1.00;
- for Selex SI, techniques filtering nouns before document indexing tend to achieve the best performances. This is consistent with a previous work by Capobianco et al. (2009). Overall, the NLP technique performing better is “Stanford nouns,TF-IDF,LSA 100,

⁵For LSA the integer after the technique, e.g., LSA 100, indicates the number of LSA concepts.

⁶<http://www.fallessi.com/EMSE-Traceability2016.zip>

Table 2 ENRL accuracy, in terms of minimum MRE, of different ML classifiers, adopting different NLP techniques, in estimating the number remaining positive links

NLP Technique	ML classifier					
	ADTree	Bagging	FLR	IBk	Logit Boost	Naive Bayes
Selex SI						
stop-stem, IDF, VSM, Jaccard	0.23	0.43	0.64	0.28	0.34	0.22
stop-stem, RAW, VSM, Dice	0.29	0.24	0.11	0.31	0.34	0.11
simple, IDF, WordNet, Pirro, Seco	0.26	0.27	0.12	0.22	0.16	0.21
simple, TF-IDF, LSA 30, cosine	0.12	0.34	0.05	0.19	0.27	0.66
Stanford nouns+verbs, IDF, VSM, Dice	0.24	0.21	0.11	0.32	0.24	0.06
Stanford nouns+verbs, IDF, WordNet, Jiang	0.39	0.29	0.10	0.33	0.42	0.11
Stanford nouns, binary, LSA 10, cosine	0.16	0.09	0.14	0.19	0.16	0.13
Stanford nouns, IDF, WordNet, Resnik	0.14	0.11	0.08	0.12	0.14	0.07
Stanford nouns, TF-IDF, VSM, cosine	0.14	0.11	0.03	0.06	0.14	0.11
Stanford nouns, TF-IDF, LSA 100, cosine	0.00	0.07	0.07	0.08	0.12	0.25
ETour						
stop-nostem, TF, LSA 100, cosine	1.00	0.99	0.99	0.18	0.82	0.19
stop-nostem, TF, LSA 50, cosine	1.00	1.00	0.70	0.03	1.00	1.00
stop-nostem, TF, VSM, cosine	0.54	0.56	0.62	0.01	0.54	0.11
stop-nostem, TF-IDF, LSA 100, cosine	1.00	1.00	1.00	0.03	1.00	1.00
stop-nostem, TF-IDF, LSA 50, cosine	0.86	0.81	0.81	0.09	0.83	0.07
stop-nostem, TF-IDF, VSM, cosine	0.92	0.36	0.87	0.09	0.23	0.09
stop-stem, TF, LSA100, cosine	1.00	1.00	1.00	0.03	1.00	1.00
stop-stem, TF, LSA50, cosine	0.60	0.60	0.68	0.04	0.60	0.05
stop-stem, TF, VSM, cosine	0.68	0.73	0.81	0.17	0.68	0.02
stop-stem, TF-IDF, LSA 100, cosine	1.00	1.00	1.00	0.03	1.00	1.00
stop-stem, TF-IDF, LSA 50, cosine	0.86	0.81	0.86	0.13	0.76	0.12
stop-stem, TF-IDF, VSM, cosine	0.62	0.35	0.85	0.05	0.60	0.10
EasyClinic						
stop-nostem, TF, LSA 100, cosine	0.09	0.06	4.47	0.13	0.12	1.00
stop-nostem, TF, LSA 50, cosine	0.23	0.23	0.64	0.03	0.19	0.02
stop-nostem, TF, VSM, cosine	0.10	0.18	0.51	0.06	0.16	0.03
stop-nostem, TF-IDF, LSA 100, cosine	0.07	0.21	8.37	0.40	0.12	1.00
stop-nostem, TF-IDF, LSA 50, cosine	0.01	0.03	0.47	0.06	0.01	0.05
stop-nostem, TF-IDF, VSM, cosine	0.22	0.22	0.45	0.06	0.21	0.04
stop-stem, TF, LSA 100, cosine	0.15	0.05	4.05	0.13	0.21	1.00
stop-stem, TF, LSA 50, cosine	0.11	0.11	0.64	0.09	0.11	0.04
stop-stem, TF, VSM, cosine	0.04	0.07	0.52	0.01	0.04	0.05
stop-stem, TF-IDF, LSA 100, cosine	0.37	0.08	8.23	0.48	0.37	1.00
stop-stem, TF-IDF, LSA 50, cosine	0.10	0.09	0.32	0.06	0.10	0.04
stop-stem, TF-IDF, VSM, cosine	0.21	0.21	0.62	0.06	0.21	0.04

- cosine” is the one providing good performances (i.e., low MRE and also passing the thresholds for all ML classifiers except Naive Bayes). In terms of ML classifier, FLR tends to exhibit better performances than others, being accurate adopting 7 out of 10 NLP techniques.
- for ETour the ML classifier IBk allows almost all NLP techniques to pass the threshold, and the best combination is obtained with “stop-stem, TF, LSA 100” (MRE = 0.03). In this case, as well as for EasyClinic discussed below, and differently from Selex SI, NLP techniques that perform pruning through part of speech analysis (i.e., selecting nouns only) were not considered among the best technique because natural language parsing works well for requirements (which were the only artifacts considered for the Selex SI systems) than for source code.
 - for EasyClinic, MRE tends to be, in general, higher than for Selex SI (maybe because in the Selex SI case vertical traceability recovery was performed), and the best combination of NLP technique and ML classifier is obtained for “stop-stem, TF, VSM” and IBk.
 - the IBk classifier performs well on the ETour and EasyClinic datasets but is outperformed by FLR in the Selex SI dataset in terms of number of accurate models. Instead, the performance of FLR is the worst in the ETour and EasyClinic datasets. One possible reason for the different performance observed in the Selex SI dataset compared to the other two datasets is that the Selex SI is an industrial dataset related to embedded systems, and therefore the quality of its requirements is likely to be reasonably higher—due to more stringent quality controls (Falessi et al. 2011)—than in open source projects. Another possible reason is the different NLP techniques used across datasets.

As stated before, the smaller the size of training set required to estimation models for providing accurate estimates, the more the ENRL is useful. Figure 4 shows, using a logarithmic scale, the accuracy of the best estimation model (i.e., minimum MRE), for each dataset, across different sizes of the training set. We prefer the use of a logarithmic scale as the difference among the observed values are large. Different estimation models required different sizes of training set to become accurate. For instance, in case of EasyClinic, the estimation model required a training set of only 20 % of the dataset to perform with an MRE of 0.01. Vice versa, the other two datasets require as training set more than 40 % of the dataset to perform with an MRE of 0.04. Although ENRL is expected to be useful before the analysts classifies 40 % of the dataset, we note that there are estimation models that require a smaller training set but perform with an MRE that is higher than the minimum.

A further important, although counter intuitive, conclusion that we can draw from Fig. 4 is that the accuracy of the estimators does not always improve while enlarging the training set. Very likely, this happens because there are links that are very difficult to trace using NLP techniques, i.e., between artifacts that are very different from a textual point of view.⁷ Thus, in most of the cases links incrementally added to the training set provide useful training to the ML classifier, other times they add noise. Moreover, the lowest MRE peak can be found in the Selex SI dataset (see Fig. 4a) where both MRE and MAE reach 0 at 75 % of the training set. Then, augmenting the training set beyond 75 % worsens results.

It is evident from Table 2 that the accuracy of the estimation model depends on both the ML classifiers and NLP techniques. We statistically tested the null hypothesis that neither the ML classifier or NLP technique influence the estimation accuracy. To test this

⁷This is a common problem in IR-based traceability link recovery (De Lucia et al. 2011).

Fig. 4 Relation between the size of the training set and MRE and MAE

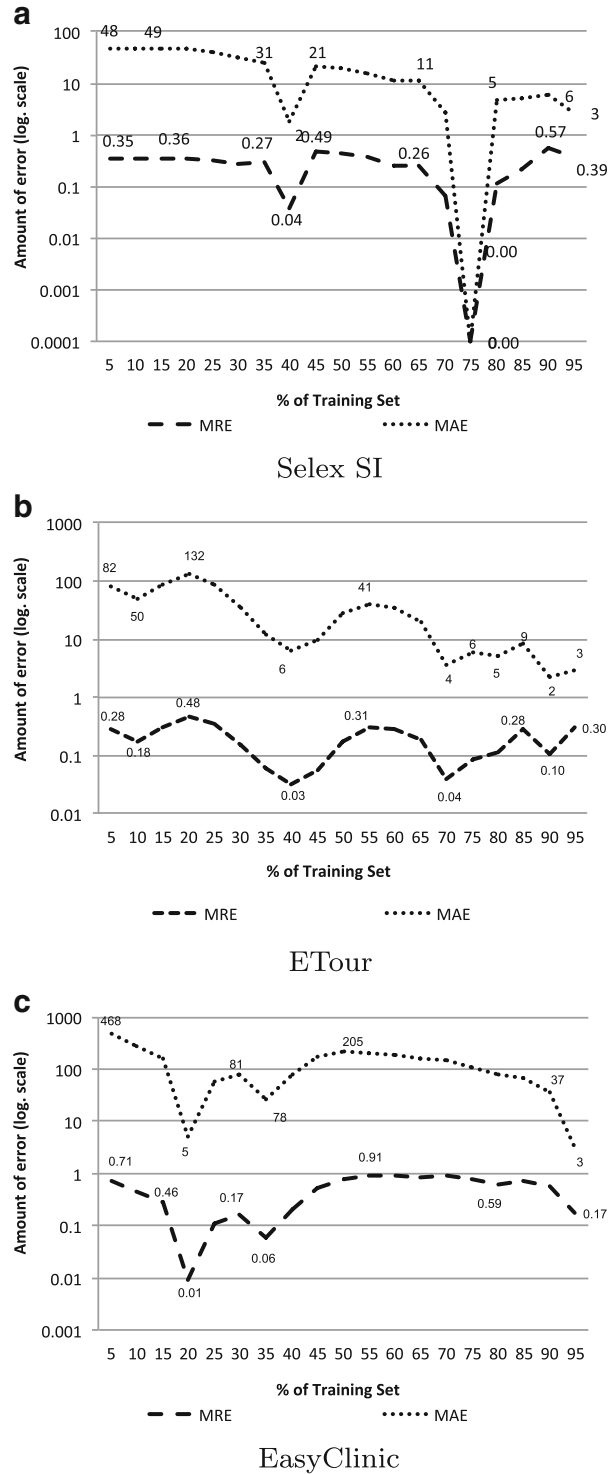


Table 3 SELEX SI - permutation test results on the influence of ML classifiers and NLP techniques on the MRE, MAE, SD, and Q90 independent variables

	Df	R Sum Sq	R Mean Sq	Iter	Pr(Prob)
MRE					
ML	7	112.10	16.01	500,000	<0.001
NLP	9	33.32	3.70	500,000	<0.001
ML:NLP	63	325.07	5.16	500,000	<0.001
Residuals	1,440	446.12	0.31		
MAE					
ML	7	275,887.05	39,412.44	500,000	<0.001
NLP	9	91,461.69	10,162.41	500,000	<0.001
ML:NLP	63	810,793.69	12,869.74	500,000	<0.001
Residuals	1,440	2,371,865.00	1,647.13		
SD					
ML	7	3.31	0.47	500,000	<0.001
NLP	9	0.35	0.04	500,000	0.07
ML:NLP	63	3.91	0.06	500,000	<0.001
Residuals	1,440	31.09	0.02		
Q90					
ML	7	142.91	20.42	500,000	<0.001
NLP	9	38.10	4.23	500,000	<0.001
ML:NLP	63	409.05	6.49	500,000	<0.001
Residuals	1440	610.90	0.42		

hypothesis, we measured accuracy as the minimum MRE observed by an estimation model over the different groups of training sets and used a permutation test (Baker 1995) to determine whether such minimum MRE depends on the NLP technique, the ML, and their interaction. The permutation test is a non-parametric alternative to the n-way Analysis of Variance (ANOVA); differently from ANOVA, it does not require data to be normally distributed. The general idea behind such a test is that the data distributions are built and compared by computing all possible values of the statistical test while rearranging the labels (representing the various factors being considered) of the data points. We used an implementation available in the *lmPerm* R package. We have set the number of iterations of the permutation test procedure to 500,000. Since the permutation test samples permutations of combination of factor levels, multiple runs of the test may produce different results. We made sure to choose a high number of iterations such that results did not vary over multiple executions of the procedure. Finally, we assumed a significance level $\alpha = 5\%$.

Tables 3, 4 and 5 report the permutation test results for SELEX SI, ETour, and EasyClinic, respectively. As we can notice, the p-values is in most cases smaller than 0.001 for all datasets and concerning both the dependence of the MRE on each factor (NLP and ML) and on their interaction (NLP:ML). There are, however, some exceptions: (i) for Selex-SI, the NLP does not have a significant effect (p-value = 0.07) on the SD dependent variable, whereas its interaction with ML does; (ii) for ETour and EasyClinic, the NLP technique and its interaction with the ML do not have a statistically significant effect on the MAE. Therefore, besides the exceptions above, we can conclude that the effect of an NLP technique on the relative error depends on the specific ML classifier adopted, and vice versa.

Table 4 ETour - permutation test results on the influence of ML classifiers and NLP techniques on the MRE, MAE, SD, and Q90 independent variables

	Df	R Sum Sq	R Mean Sq	Iter	Pr(Prob)
MRE					
ML	6	97.19	16.20	500,000	< 0.001
NLP	11	4.78	0.43	500,000	< 0.001
ML:NLP	66	16.08	0.24	500,000	< 0.001
Residuals	1,512	21.67	0.01		
MAE					
ML	6	2,620,856.28	436,809.38	500,000	< 0.001
NLP	11	143,285.65	13,025.97	500,000	0.06
ML:NLP	66	473,007.42	7,166.78	500,000	0.56
Residuals	1,512	11,157,314.20	7379.18		
SD					
ML	6	0.49	0.08	500,000	< 0.001
NLP	11	0.24	0.02	500,000	< 0.001
ML:NLP	66	0.26	0.00	500,000	< 0.001
Residuals	1,512	2.27	0.00		
Q90					
ML	6	84.89	14.15	500,000	< 0.001
NLP	11	3.30	0.30	500,000	< 0.001
ML:NLP	66	13.98	0.21	500,000	< 0.001
Residuals	1,512	21.39	0.01		

Let us consider, for example, for Selex SI (Table 2-a) the NLP techniques “Stanford nouns and verb , IDF, VSM, Dice” (row 5) and “Stanford nouns, TF-IDF, LSA 100, cosine” (row 9): while the former performs well if used with Naive Bayes (min MRE = 0.06), and it does not perform well with ADTree (min MRE = 0.24), the latter NLP technique does not pass the threshold with Naive Bayes (min MRE = 0.25) while it performs very well with ADTree (min MRE = 0.00). In conclusion, the level of support of an NLP technique to ENRL depends on the specific ML classifier it is used with.

Results above showed that the estimation accuracy changes among ML classifiers, datasets, and NLP techniques in use. This would suggest a high variation of accuracy among ML classifiers. We now investigate if the accuracy changes among datasets and ML classifiers when adopting an ideal perfect NLP technique. The ideal perfect NLP technique is defined as a technique providing a similarity score of 1 in case of a true link, and 0 otherwise. Therefore, we computed the accuracy of ML classifiers according to the procedure reported in Algorithm 1 having as similarity score (line 13) a perfect ideal score. According to Table 6, all ML classifiers, other than Bagging, reach the best ideal accuracy (i.e., no error) when adopting the best ideal NLP technique. Thus, the choice of the ML classifier to use is almost irrelevant when using a perfect NLP technique.

We note that a model required few minutes to run in a standard laptop and no significant differences were observed among different models. Therefore, the time required to run the model should not influence its selection.

Table 5 EasyClinic - permutation test results on the influence of ML classifiers and NLP techniques on the MRE, MAE, SD, and Q90 independent variables

	Df	R Sum Sq	R Mean Sq	Iter	Pr(Prob)
MRE					
ML	6	41,076.71	6,846.12	500,000	<0.001
NLP	11	246.30	22.39	500,000	<0.001
ML:NLP	66	590.69	8.95	500,000	<0.001
Residuals	1,512	7,052.04	4.66		
MAE					
ML	6	2,775,246,663.37	462,541,110.56	500,000	<0.001
NLP	11	43,301,088.25	3,936,462.57	500,000	<0.001
ML:NLP	66	119,147,336.56	1,805,262.68	500,000	<0.001
Residuals	1,512	1,098,778,187.55	726,705.15		
SD					
ML	6	90.92	15.15	500,000	<0.001
NLP	11	2.01	0.18	500,000	0.15
ML:NLP	66	2.65	0.04	500,000	1.00
Residuals	1,512	193.43	0.13		
Q90					
ML	6	45,144.83	7524.14	500,000	<0.001
NLP	11	309.54	28.14	500,000	<0.001
ML:NLP	66	620.16	9.40	500,000	<0.001
Residuals	1,512	7,616.06	5.04		

RQ1 Summary: Several estimation models provide accurate estimates about the number of remaining positive links. Moreover, their accuracy does not necessarily improve when increasing the training set size. Thus, the ENRL approach can be effectively used by the analyst to make informed decision about when to stop searching for artifacts to link.

Table 6 Accuracy of ML classifiers adopting an ideal NLP technique

ML	SeleX SI				EasyClinic				ETour			
	MRE	SD	90Q	MAE	MRE	SD	90Q	MAE	MRE	SD	90Q	MAE
ADTree	0	0	0	0	0	0	0	0	0	0	0	0
Bagging	0.21	0.41	1	28.5	0	0	0	0	0.014	0.12	0	4.27
FLR	0	0	0	0	0	0	0	0	0	0	0	0
IBk	0	0	0	0	0	0	0	0	0	0	0	0
LogitBoost	0	0	0	0	0	0	0	0	0	0	0	0
NaïveBayes	0	0	0	0	0	0	0	0	0	0	0	0
ZeroR	1	0	1	71.25	1	0	1	336.83	1	0	1	165.07

5.2 RQ2: Can we Select Accurate Estimation Models?

As discussed in Section 4.5, in order to answer this research question we analyze the possibility to estimate the accuracy of estimation models, thus we will analyze the correlation between the actual MRE and the estimated MRE (computed via 10-fold cross validation).

We analyze correlation both descriptively and inferentially. Regarding the descriptive analysis, we make use of scatter-plots, where the x-axis represent the estimator variable and the y-axis the estimated variable. In our case, we want to estimate the actual MRE. Moreover, a default 95 % bivariate normal density ellipse is shown in each scatter-plot. The narrowness of the ellipse shows the correlation of the variables. If the ellipse is fairly round and is not diagonally oriented, the variables are uncorrelated. If the ellipse is narrow and diagonally oriented, the variables are correlated.

Figure 5 shows the scatter-plot of estimated (x-axis) versus actual (y-axis) MRE, in each of the three datasets, for all the 60 estimation models considered in this study. According to Fig. 5, the correlation is clearly high in both ETour and EasyClinic data sets (Fig. 5b and c) and it is small in the SelexSI dataset (Fig. 5a).

Regarding inferential analysis, we measured the Spearman and Kendall coefficients for the three datasets considered in this work. We avoided the use of the Pearson correlation

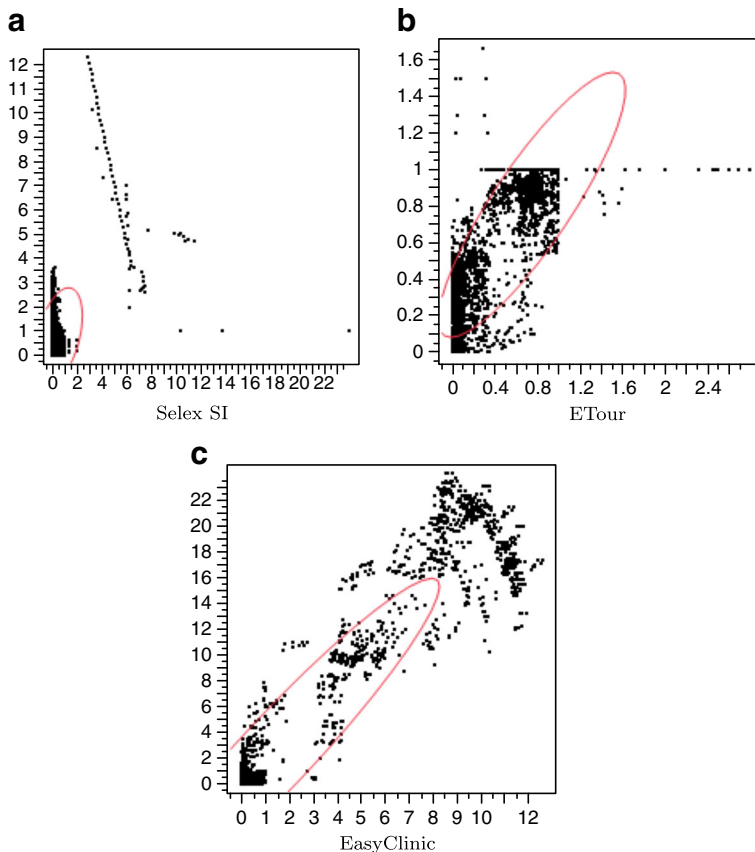


Fig. 5 Scatterplot analyzing the relation between estimated (x-axis) and actual (y-axis) MRE

Table 7 Correlation between estimated and actual MRE

Data set	Spearman		Kendall	
	ρ	p-value	τ	p-value
Selex SI	0.046	< 0.0001	0.035	< 0.0001
ETour	0.86	< 0.0001	0.74	< 0.0001
EasyClinic	0.55	< 0.0001	0.40	< 0.0001

as it requires the two variables to be approximately normally distributed and their relation to be linear without considering further monotonic relations. Since such assumptions were not met by our data we opted to use two non-parametric correlation coefficients: Spearman and Kendall. In general, these two non-parametric coefficients usually produce very similar results, though Spearman is usually higher than Kendall, and there is no strong reason for preferring one over the other (Colwell and Gillett 1982); thus, we used both of them to provide a comprehensive analysis. We adopted the same interpretation procedure. Specifically, both coefficients range from -1 to $+1$. The sign of the correlation indicates the direction of association between the two variables; a positive value means that when one variable increases the other variable does as well, a negative value means that when one variable increases the other variable decreases. The absolute value describes the strength of the relation; when two variables are independent then the coefficients are approximately 0, whereas when two variables are a perfect function of the other the coefficients are -1 or $+1$ (Myers and Well 2003). We also measured the p-value which represents the likelihood of the observed correlation to be caused by chance.

Table 7 reports the values of the Spearman and Kendall coefficients and their p-values when correlating the estimated with the actual MRE. According to the table, the correlation is clearly high in both ETour and EasyClinic datasets (columns 2 and 4, rows 4 and 5) and it is small in the SelexSI dataset (columns 2 and 4, row 3). This result is consistent with the qualitative analysis in Fig. 5. Moreover, according to Table 7 all p-values resulted very low and thus we can have a high confidence that the observed correlation between the estimated MRE and the actual MRE truly exists, rather than caused by chance.

RQ2 Summary: The estimated MRE, as computed via ten-fold validation, seems to be a good indicator to select an accurate estimation model.

5.3 RQ3:Do Multivariate Estimation Models Outperform Univariate Ones?

Table 8 reports, for each of the three datasets, the accuracy, in terms of (actual) MRE of each of the six ML classifiers when using the similarity scores provided by all NLP techniques (MV), the similarity scores provided by the actual best NLP technique (ABS), or the similarity scores provided by the NLP technique the user would choose because having the highest estimated accuracy (EBS). Note that the MRE in Table 8 is in average among the 95 training/test sets whereas the MRE in Table 2 is the minimum among the training/test sets.

According to Table 8, for all three datasets the use of the actual best NLP technique is preferable to the use of all NLP techniques together, for 5 out of 6 ML classifiers (i.e., all ML classifiers except for FLR and ZeroR). Moreover, for the ETour and EasyClinic datasets, for all the 6 ML classifiers, the use of the estimated best NLP technique is preferable to the

Table 8 Accuracy (MRE) of the six ML classifiers (row) across different datasets and different use of NLP techniques (column)

ML Classifier	Selex			ETour			EasyClinic		
	MV	ABS	EBS	MV	ABS	EBS	MV	ABS	EBS
ADTree	0.313	0.290	0.672	1.000	0.914	0.932	0.811	0.484	0.574
Bagging	0.523	0.485	0.936	1.000	0.785	0.909	0.641	0.446	0.569
FLR	1.000	0.220	0.263	1.000	0.817	0.817	1.000	12.290	12.290
IBk	1.000	0.278	0.432	1.000	0.151	0.333	1.000	0.406	0.582
LogitBoost	0.318	0.296	0.793	0.985	0.785	0.895	0.827	0.495	0.577
NaïveBayes	1.000	0.526	1.159	1.000	0.362	0.401	1.000	0.465	0.486

MV = Multivariate; ABS = Actual Single NLP technique; EBS = Estimated Best Single NLP technique

use of all NLP techniques together. Regarding the SelexSI dataset, we have this result only on 2 ML classifiers out of 6.

Regarding the highest accuracy, in Table 8 we highlighted in gray the accuracy of the best ML classifier. Results show that in all three datasets, using the expected best NLP technique is always better than using all NLP techniques; i.e., EBS (third column) has a lower MRE than MV (first column) in all three datasets (subtables of Table 8). The achieved results suggest the use of an estimation model adopting the (expected) best NLP technique rather than adopting all NLP techniques. It also suggest the investigation of better ways to aggregate several NLP techniques together compared to just their simple aggregation. One suitable option is filtering the NLP techniques before their aggregation by applying, as already done for similar aims (Falessi et al. 2011; Nagappan et al. 2006), the principal component analysis.

RQ3 Summary: The use of several NLP techniques simultaneously does not improve the estimation accuracy over using one NLP technique at a time.

6 Threats to Validity

This section describes the threats to validity that can affect the empirical study presented in this paper.

Threats to *construct validity* concern the relationship between theory and observation (Wohlin et al. 2000). They are often related to whether the measurements performed in the experimental settings are realistic and accurate. For both ML classifiers and NLP techniques, we used widely adopted performance measures, i.e., relative errors and its statistics for the classifier, accuracy, precision, and recall for NLP techniques. Due to feasibility constraints (i.e., 12 NLP techniques) we were limited on how many values of K we could adopt when using the LSA alternative in the NLP technique. Therefore, we chosen K according to the size of the dataset. This unfortunately led to the use of different NLP techniques in different datasets. Nevertheless, we believe that further studies on other NLP techniques and ML classifiers are desired to confirm or contradict our results.

Threats to *conclusion validity* concern the relationship between treatment and outcome (Wohlin et al. 2000). Not only our results are obtained using a very large data set of

candidate links, but, also, wherever possible, we support our finding with appropriate statistical tests, such as the F-test for **RQ1**.

Threats to *internal validity* concern factors that could affect our results but that we did not consider (Wohlin et al. 2000). First, we are aware that Weka ML classifiers were used out-of-the box, without a proper calibration. This is because we wanted to assess their performance in the worst-case-scenario where the user just applies them without any attempt of tuning. Indeed, a proper calibration could lead to even better results. Specifically we expect that using calibrated estimation models would result in higher accuracy when trained with the same amount of data, and same accuracy when trained with a smaller amount data. Moreover, for facilitating replicas, we made ATHLETE open-source and available for download. Another threat to internal validity could be that the performance of a ML classifier does not depend on the choice of the NLP technique, but, rather, it may depend on other factors, for example the particular characteristics of the adopted textual corpus. Finally, one possible factor that could have influenced our results is the choice of thresholds used to evaluate the approach's accuracy in RQ1. In Section 4.5.1, we have discussed how such thresholds have been set.

Threats to *external validity* are related to the generalizability of the obtained results (Wohlin et al. 2000). We considered both a large industrial dataset (Selex SI) and two medium-small publicly available datasets from academia (ETour and EasyClinic). Also, such dataset involve both horizontal and vertical traceability link recovery between different kinds of software artifacts. Nevertheless, we believe that further studies on other datasets are desired to confirm or contradict our results. We have experimented a large set of ML classifiers (7) available in Weka and NLP techniques (12). Although we are aware that techniques we did not consider could exhibit different results from those we have reported, the total number of considered estimation models is 238 (i.e., 7 ML classifiers * 12 NLP techniques * 3 datasets) which is a very high number. The differences in the adopted NLP techniques are due to the different characteristics of the datasets. Finally, it is important to point out that we did not apply exactly the same set of NLP techniques to all datasets. Although we applied every time 12 NLP techniques, we cannot exclude that some techniques, not applied on one specific dataset, would have produced particularly high (or low) performances in that case.

7 Conclusions

This paper proposes a new approach called ENRL which aims at estimating, via ML classifiers, the number of remaining positive links in a ranked list of candidate traceability links produced by an NLP-based recovery approach. The ENRL approach provides objective informative data to any requirements analyst for deciding to stop recovering traceability links when following a ranked list of candidates. We did not found any other approach supporting analysts in such a scenario. We have evaluated the accuracy of the ENRL approach by considering seven ML classifiers and several (12 for each dataset) NLP techniques on three datasets from both industry and academia, and concerning traceability links among different kinds of software artifacts including requirements, use case, design documents, source code, and test cases.

Results of the study show that:

- *RQ1: What is the level of accuracy of estimation models?* Several estimation models provide accurate estimates (i.e., $MRE < 0.15$, $SD < 0.15$, $90Q < 0.30$, and $MAE <$

0.05) about the number of remaining positive links, by just using out-of-the box NLP techniques and ML classifiers. Thus, the ENRL approach can be effectively used by the analyst to make informed decision about when to stop searching for artifacts to link. However, it seems that the accuracy of estimation models differs among datasets, and there is no estimation model dominating the others on all datasets. This result is in line with results on traceability recovery where the level of support of NLP techniques in retrieving positive links varies hugely among contexts and datasets (Gethers et al. 2011; De Lucia et al. 2011). Finally, two estimation models resulted accurate in all three datasets. Therefore, the analyst can chose the estimation model according to our results or can execute the methodology provided in pseudo code and code to measure which model is the most accurate with the dataset under analysis.

- *RQ2: Can we select accurate estimation models?* The estimated accuracy resulted to be highly correlated with the actual accuracy. Therefore, in a situation where some estimation models are accurate and some are not, the estimated accuracy is a reliable criteria to chose estimation model.
- *RQ3: Do multivariate estimation models outperform univariate ones?* It is clearly better to use the NLP technique that is expected to have a high accuracy rather than using several NLP techniques simultaneously. This is true for most of the ML classifiers and for the ML classifier showing the highest estimation accuracy in each of the three datasets.

In conclusion, this paper presented the first exploratory investigation about the possibility to estimate the number of remaining links adopting ML classifiers and NLP techniques. In contexts where analysts need to manually inspect the set of candidate links between two software artifacts—e.g., between a requirement and other related requirements, or between a requirement and source code files, in the context of a software maintenance activity—the estimation of the number of remaining candidate links can help the analyst deciding when to stop in such a manual analysis, and to properly plan her activity accordingly.

The achieved results motivate further research efforts and thus they pave the way to a high number of research investigations. The ENRL approach is based on the use of supervised classifiers, though in the future we could investigate the use of unsupervised classifiers, e.g., by estimating the similarity threshold used to count the number of remaining links without using labeled data, as it has been recently done for defect prediction (Nam and Kim 2015). In the future we also plan to replicate this study over a larger spectrum of datasets, NLP techniques and ML classifiers. Moreover, we plan to investigate further factors which could influence the possibility to use ENRL including analysts' experience and context characteristics. Additionally, we plan to replicate the study by using other datasets or other combinations of ML classifiers and NLP techniques. Finally, we plan to perform controlled experiments aimed at investigating the actual benefits analysts have by using ENRL.

References

- Abadi A, Nisenson M, Simionovici Y (2008) A traceability technique for specifications. In: The 16th IEEE international conference on program comprehension, ICPC 2008, Amsterdam, The Netherlands, June 10–13, 2008. IEEE CS, pp 103–112
- Altman NS (1992) An introduction to kernel and nearest-neighbor nonparametric regression. *Am Stat* 46(3):175–185
- Antoniol G, Canfora G, Casazza G, De Lucia A (2000) Identifying the starting impact set of a maintenance request: a case study. In: European conference on software maintenance and reengineering, CSMR, pp 227–230
- Antoniol G, Canfora G, Casazza G, De Lucia A, Merlo E (2002) Recovering traceability links between code and documentation. *IEEE Trans Softw Eng* 28(10):970–983

- Asuncion HU, Asuncion AU, Taylor RN (2010) Software traceability with topic modeling. In: Proceedings of the 32nd ACM/IEEE international conference on software engineering - volume 1, ICSE 2010, Cape Town, South Africa, 1–8 May 2010, pp 95–104
- Athanasiadis I (2007) The fuzzy lattice reasoning (flr) classifier for mining environmental data. In: Kaburlasos V, Ritter G (eds) Computational intelligence based on lattice theory, studies in computational intelligence, vol 67. Springer, Berlin, Heidelberg, pp 175–193. doi:[10.1007/978-3-540-72687-6_9](https://doi.org/10.1007/978-3-540-72687-6_9)
- Baeza-Yates R, Ribeiro-Neto B (1999) Modern information retrieval. Addison-Wesley
- Bai CG, Cai KY, Hu QP, Ng SH (2008) On the trend of remaining software defect estimation. IEEE Trans Syst Man Cybern Part A Syst Humans 38(5):1129–1142. doi:[10.1109/TSMCA.2008.2001071](https://doi.org/10.1109/TSMCA.2008.2001071)
- Baker RDEdgington E (ed) (1995) Modern permutation test software. Marcel Decker
- Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3:993–1022. doi:[10.1162/jmlr.2003.3.4-5.993](https://doi.org/10.1162/jmlr.2003.3.4-5.993)
- Borg M, Runeson P, Ardö A (2014) Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability. Empir Softw Eng 19(6):1565–1616. doi:[10.1007/s10664-013-9255-y](https://doi.org/10.1007/s10664-013-9255-y)
- Breiman L, Breiman L (1996) Bagging predictors. In: Machine learning, pp 123–140
- Briand LC, Emam KE, Freimut BG, Laitenberger O (2000) A comprehensive evaluation of capture-recapture models for estimating software defect content. IEEE Trans Softw Eng 26(6):518–540
- Briand LC, Falessi D, Nejati S, Sabetzadeh M, Yue T (2014) Traceability and SysML design slices to support safety inspections: a controlled experiment. ACM Trans Softw Eng Methodol 23(1):9:1–9:43. doi:[10.1145/2559978](https://doi.org/10.1145/2559978)
- Cai K (1998) On estimating the number of defects remaining in software. J Syst Softw 40(2):93–114. doi:[10.1016/S0164-1212\(97\)00003-4](https://doi.org/10.1016/S0164-1212(97)00003-4)
- Capobianco G, De Lucia A, Oliveto R, Panichella A, Panichella S (2009) On the role of the nouns in IR-based traceability recovery. In: The 17th IEEE international conference on program comprehension, ICPC 2009, Vancouver, British Columbia, Canada, May 17–19, 2009. IEEE CS, pp 148–157
- Chen T, Sahinoglu M, von Mayrhauser A, Hajjar A, Anderson C (1999) How much testing is enough? Applying stopping rules to behavioral model testing. In: 4th IEEE international symposium on high-assurance systems engineering, 1999. Proceedings, pp 249–256. doi:[10.1109/HASE.1999.809500](https://doi.org/10.1109/HASE.1999.809500)
- Cleland-Huang J, Settini R, Duan C, Zou X (2005) Utilizing supporting evidence to improve dynamic requirements traceability. In: 13th IEEE international conference on requirements engineering (RE 2005), 29 August - 2 September 2005, Paris, France. IEEE CS, pp 135–144
- Colwell DJ, Gillett JR (1982) 66.49 Spearman versus Kendall. Math Gaz 66(438):307–309
- Cover TM, Thomas JA (1991) Elements of information theory. Wiley-Interscience
- Cuddeback D, Dekhtyar A, Huffman Hayes J, Holden J, Kong W-K (2011) Towards overcoming human analyst fallibility in the requirements tracing process. In: Proceedings of the 33rd international conference on software engineering, ICSE 2011, Waikiki, Honolulu, HI, USA, May 21–28, 2011. ACM, pp 860–863
- Czadurna A, Cleland-Huang J, Cinar M, Berenbach B (2012) Just-in-time traceability for mechatronics systems. In: IEEE second workshop on requirements engineering for systems, services and systems-of-systems (RES4), 2012, pp 1–9. doi:[10.1109/RES4.2012.6347691](https://doi.org/10.1109/RES4.2012.6347691)
- Dag JN, Regnell B, Carlshamre P, Andersson M, Karlsson J (2002) A feasibility study of automated natural language requirements analysis in market-driven development. Requir Eng 7(1):20–33
- De Lucia A, Oliveto R, Sgueglia P (2006) Incremental approach and user feedbacks: a silver bullet for traceability recovery. In: 22nd IEEE international conference on software maintenance (ICSM 2006), 24–27 September 2006, Philadelphia, Pennsylvania, USA. IEEE Computer Society, pp 299–309
- De Lucia A, Fasano F, Oliveto R, Tortora G (2007) Recovering traceability links in software artifact management systems using information retrieval methods. ACM Trans Softw Eng Methodol 16(4)
- De Lucia A, Oliveto R, Tortora G (2009) Assessing IR-based traceability recovery tools through controlled experiments. Empir Softw Eng 14(1):57–92
- De Lucia A, Di Penta M, Oliveto R, Panichella A, Panichella S (2011) Improving IR-based traceability recovery using smoothing filters. In: The 19th IEEE international conference on program comprehension, ICPC 2011, Kingston, ON, Canada, June 22–24, 2011. IEEE Computer Society, pp 21–30
- Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R (1990) Indexing by latent semantic analysis. J Am Soc Inf Sci 41(6):391–407
- Dekhtyar A, Dekhtyar O, Holden J, Hayes JH, Cuddeback D, Kong W-K (2011) On human analyst performance in assisted requirements tracing: statistical analysis. In: RE 2011, 19th IEEE international requirements engineering conference, Trento, Italy, August 29 2011–September 2, 2011. IEEE, pp 111–120
- Duan C, Cleland-Huang J (2007) Clustering support for automated tracing. In: 22nd IEEE/ACM international conference on automated software engineering (ASE 2007), November 5–9, 2007, Atlanta, Georgia, USA. ACM, pp 244–253

- Falessi D, Reichel A (2015) Towards an open-source tool for measuring and visualizing the interest of technical debt. In: IEEE 7th international workshop on managing technical debt (MTD), 2015, pp 1–8. doi:[10.1109/MTD.2015.7332618](https://doi.org/10.1109/MTD.2015.7332618)
- Falessi D, Briand LC, Cantone G (2009) The impact of automated support for linking equivalent requirements based on similarity measures. Tech. rep., Simula Research Laboratory Technical Report 2009–08
- Falessi D, Cantone G, Canfora G (2011) Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. *IEEE Trans Softw Eng* 39(1):18–44
- Falessi D, Shaw MA, Mullen K (2014) Achieving and maintaining CMMI maturity level 5 in a small organization. *IEEE Softw* 31(5):80–86. doi:[10.1109/MS.2014.17](https://doi.org/10.1109/MS.2014.17)
- Fellbaum C (1998) Wordnet: an electronic lexical database. The MIT Press
- Foss T, Stensrud E, Kitchenham B, Myrteit I (2003) A simulation study of the model evaluation criterion MMRE. *IEEE Trans Softw Eng* 29(11):985–995
- Freund Y, Mason L (1999) The alternating decision tree learning algorithm. In: In machine learning: proceedings of the sixteenth international conference. Morgan Kaufmann, pp 124–133
- Friedman J, Hastie T, Tibshirani R (2000) Additive logistic regression: a statistical view of boosting. *Ann Stat* 28:1998
- Gethers M, Oliveto R, Poshvanyk D, De Lucia A (2011) On integrating orthogonal information retrieval methods to improve traceability recovery. In: IEEE 27th international conference on software maintenance, ICSM 2011, Williamsburg, VA, USA, September 25–30, 2011. IEEE, pp 133–142
- Hayes JH, Dekhtyar A, Sundaram SK (2006) Advancing candidate link generation for requirements tracing: the study of methods. *IEEE Trans Softw Eng* 32(1):4–19
- Huffman Hayes J, Dekhtyar A, Osborne J (2003) Improving requirements tracing via information retrieval. In: 11th IEEE international conference on requirements engineering (RE 2003), 8–12 September 2003, Monterey Bay, CA, USA. IEEE CS, p 138
- Kim S, Zhang H, Wu R, Gong L (2011) Dealing with noise in defect prediction. In: Proceedings of the 33rd international conference on software engineering, ICSE '11. ACM, New York, NY, USA, pp 481–490. doi:[10.1145/1985793.1985859](https://doi.org/10.1145/1985793.1985859)
- Krishnan S, Strasburg C, Lutz RR, Goseva-Popstojanova K, Dorman KS (2013) Predicting failure-proneness in an evolving software product line. *Inf Softw Technol* 55(8):1479–1495. doi:[10.1016/j.infsof.2012.11.008](https://doi.org/10.1016/j.infsof.2012.11.008)
- Lindvall M, Sandahl K (1996) Practical implications of traceability. *Softw Pract Exper* 26(10):1161–1180. doi:[10.1002/\(SICI\)1097-024X\(199610\)26:10<1161::AID-SPE58>3.3.CO;2-O](https://doi.org/10.1002/(SICI)1097-024X(199610)26:10<1161::AID-SPE58>3.3.CO;2-O)
- Lohar S, Amornborvornwong S, Zisman A, Cleland-Huang J (2013) Improving trace accuracy through data-driven configuration and composition of tracing features. In: Joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, ESEC/FSE'13, Saint Petersburg, Russian Federation, August 18–26, 2013. ACM, pp 378–388
- Lormans M, van Deursen A (2006) Can Isi help reconstructing requirements traceability in design and test? In: 10th European conference on software maintenance and reengineering (CSMR 2006), 22–24 March 2006, Bari, Italy. IEEE Computer Society, pp 47–56
- Lormans M, van Deursen A, Groß H (2008) An industrial case study in reconstructing requirements views. *Empir Softw Eng* 13(6):727–760. doi:[10.1007/s10664-008-9078-4](https://doi.org/10.1007/s10664-008-9078-4)
- Malaiya YK, Denton J (1998) Estimating the number of residual defects [in software]. In: High-assurance systems engineering symposium, 1998. Proceedings. Third IEEE international, pp 98–105. doi:[10.1109/HASE.1998.731600](https://doi.org/10.1109/HASE.1998.731600)
- Marcus A, Maletic JI (2003) Recovering documentation-to-source-code traceability links using latent semantic indexing. In: Proceedings of the 25th international conference on software engineering, May 3–10, 2003, Portland, Oregon, USA. IEEE CS, pp 125–137
- Mirakhorli M, Cleland-Huang J (2011) Tracing architectural concerns in high assurance systems (niet track). In: Proceedings of the 33rd international conference on software engineering, ICSE '11. ACM, New York, NY, USA, pp 908–911. doi:[10.1145/1985793.1985942](https://doi.org/10.1145/1985793.1985942)
- Mirakhorli M, Shin Y, Cleland-Huang J, Cinar M (2012) A tactic-centric approach for automating traceability of quality concerns. In: 2012 34th international conference on software engineering (ICSE), pp 639–649. doi:[10.1109/ICSE.2012.6227153](https://doi.org/10.1109/ICSE.2012.6227153)
- Myers JL, Well AD (2003) Research design and statistical analysis. Lawrence Erlbaum Associates, New Jersey
- Nagappan N, Ball T, Zeller A (2006) Mining metrics to predict component failures. In: Proceedings of the 28th international conference on software engineering, ICSE '06. ACM, New York, NY, USA, pp 452–461. doi:[10.1145/1134285.1134349](https://doi.org/10.1145/1134285.1134349)
- Nam J, Kim S (2015) Clami: defect prediction on unlabeled datasets. In: Proceedings of the 30th IEEE/ACM international conference on automated software engineering (ASE 2015)
- Okutan A, Yildiz OT (2014) Software defect prediction using bayesian networks. *Empir Softw Eng* 19(1):154–181. doi:[10.1007/s10664-012-9218-8](https://doi.org/10.1007/s10664-012-9218-8)

- Otis D, Burnham K, White G, Andersonm D (1978) Statistical inference from capture data on closed animal population. *Wildl Monogr* 62(135)
- Panichella A, Dit B, Oliveto R, Di Penta M, Poshyanyk D, Lucia AD (2013) How to effectively use topic models for software engineering tasks? An approach based on genetic algorithms. In: 35th international conference on software engineering, ICSE '13, San Francisco, CA, USA, May 18–26, 2013. IEEE/ACM, pp 522–531
- Petersson H, Thelin T, Runeson P, Wohlin C (2004) Capture-recapture in software inspections after 10 years research—theory, evaluation and application. *J Syst Softw* 72(2):249–264
- Porter MF (1980) An algorithm for suffix stripping. *Program* 14(3):130–137
- Rahman F, Posnett D, Herraiz I, Devanbu P (2013) Sample size vs. bias in defect prediction. In: Proceedings of the 2013 9th joint meeting on foundations of software engineering, ESEC/FSE 2013. ACM, New York, NY, USA, pp 147–157. doi:[10.1145/2491411.2491418](https://doi.org/10.1145/2491411.2491418)
- Russell SJ, Norvig P (2003) Artificial intelligence: a modern approach, 2nd edn. Pearson Education
- Settimi R, Cleland-Huang J, Khadra OB, Mody J, Lukasik W, DePalma C (2004) Supporting software evolution through dynamically retrieving traces to UML artifacts. In: 7th international workshop on principles of software evolution (IWPSE 2004), 6–7 September 2004, Kyoto, Japan. IEEE Computer Society, pp 49–54
- Stone M (1974) Cross-validators choice and assesment of statistical predictions (with discussion). *J R Stat Soc Ser B* 36:111–147
- Witten IH, Frank E, Hall MA (2011) Data mining: practical machine learning tools and techniques, 3rd edn. Morgan Kaufmann Publishers Inc., San Francisco
- Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2000) Experimentation in software engineering: an introduction. Kluwer Academic Publishers, Norwell
- Yadla S, Huffman Hayes J, Dekhtyar A (2005) Tracing requirements to defect reports: an application of information retrieval techniques. *ISSE* 1(2):116–124
- Zou X, Settimi R, Cleland-Huang J (2007) Term-based enhancement factors for improving automated requirement trace retrieval. In: ACM international symposium on grand challenges of traceability
- Zou X, Settimi R, Cleland-Huang J (2010) Improving automated requirements trace retrieval: a study of term-based enhancement methods. *Empir Softw Eng* 15(2):119–146



Davide Falessi is an associate professor at the California Polytechnic State University, in San Luis Obispo, USA. His main research interest is in devising and empirically assessing scalable solutions for the development of software-intensive systems with a particular emphasis on technical debt, traceability, and architecture. He is the first author of numerous papers appeared in the most influential international journals in software engineering including CSUR, EMSE, IEEE Software, IST, JSS, TSE, and TOSEM. He serves and has served in the organizing and program committees of more than 50 conferences such as ESEM, ICSE, ICSE-SEIP, and others. He is the guest co-editor of the special issue on “Technical Debt in Software Systems” recently published in the *Journal of Systems and Software*, Elsevier. He is also the guest co-editor of an upcoming special issue on “Actionable Analytics for Software Engineering” to be published in the *IEEE Software*. He is the Associate Editor in Software Economics, and the Multimedia Editor, of *IEEE Software*. He is a member at large of IEEE Computer Society publications board.



Massimiliano Di Penta is an associate professor at the University of Sannio, Italy. His research interests include software maintenance and evolution, mining software repositories, empirical software engineering, search-based software engineering, and service-centric software engineering. He is an author of more than 230 papers appeared in international journals, conferences, and workshops. He serves and has served in the organizing and program committees of more than 100 conferences such as ICSE, FSE, ASE, ICSM, ICPC, GECCO, MSR WCRE, and others. He has been a general co-chair of various events, including the 10th IEEE Working Conference on Source Code Analysis and Manipulation (SCAM 2010), the second International Symposium on Search-Based Software Engineering (SSBSE 2010), and the 15th Working Conference on Reverse Engineering (WCRE 2008). Also, he has been a program chair of events such as the 28th IEEE International Conference on Software Maintenance (ICSM 2012), the 21st IEEE International Conference on Program Comprehension (ICPC 2013), the ninth and 10th Working Conference on Mining Software Repository (MSR 2013 and 2012), the 13th and 14th Working Conference on Reverse Engineering (WCRE 2006 and 2007), the first International Symposium on Search-Based Software Engineering (SSBSE 2009), and other workshops. He is currently a member of the steering committee of ICSME, MSR, SSBSE, and PROMISE. Previously, he has been a steering committee member of other conferences, including ICPC, SCAM, and WCRE. He is in the editorial board of the IEEE Transactions on Software Engineering, the Empirical Software Engineering Journal edited by Springer, and of the Journal of Software: Evolution and Processes edited by Wiley.



Gerardo Canfora is a professor of computer science at the Faculty of Engineering of the University of Sannio, Italy. He serves on the program and organizing committees of a number of international conferences. He was general chair of WCRE'06 and CSMR'03, and program co-chair of ICSE'15, WETSoM'12 and '10, ICSM'01 and '07, IWPSE'05, CSMR'04 and IWPC'97. He is co-editor of the "Journal of Software: Evolution and Processes". Canfora authored 200 research papers; his research interests include software maintenance and evolution, security and privacy, empirical software engineering, and service-oriented computing.



Giovanni Cantone is Professor of Software Systems Engineering in the Department of Civil Engineering and Computer Science of the University of Roma “Tor Vergata”, Italy. Formerly, he was an Associate professor in the University of Napoli “Frederic the 2nd”, Italy, and Research associate in the DCS of the University of Maryland, USA. He is a founding member of ISERN, and he was general chair of ESEIW 2003, and program co-chair of ISESE 2003 and XP 2014. He has been serving in the program committee of a number of international conferences. Cantone authored more than one hundred research papers; his research interests include Software architecture, Software metrics, Software quality, Software organization management, and Empirical software engineering.