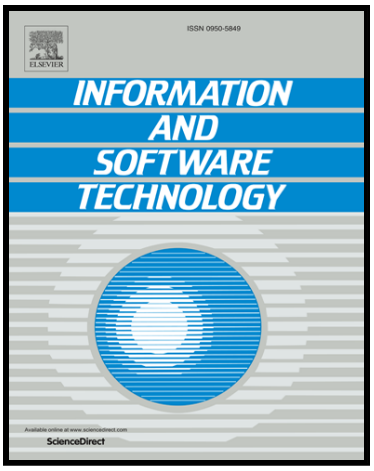


Exploiting Parts-of-Speech for Effective Automated Requirements Traceability

Nasir Ali, Haipeng Cai, Abdelwahab Hamou-Lhadj, Jameleddine Hassine

PII: S0950-5849(18)30204-0
DOI: https://doi.org/10.1016/j.infsof.2018.09.009
Reference: INFSOF 6056

To appear in: *Information and Software Technology*

Received date: 7 November 2017
Revised date: 10 September 2018
Accepted date: 25 September 2018

Exploiting Parts-of-Speech for Effective Automated Requirements Traceability

Nasir Ali[a], Haipeng Cai[b], Abdelwahab Hamou-Lhadj[c,*], Jameleddine Hassine[d]

[a]*Department of Computer Science, University of Memphis, Tennessee, USA*
[b]*School of Electrical Engineering and Computer Science, Washington State University, Pullman, Washington, USA*
[c]*Electrical and Computer Engineering Department, Concordia University, Montréal, Canada*
[d]*Department of Information and Computer Science, King Fahd University of Petroleum and Minerals, Dhahran, KSA*

Abstract

Context: Requirement traceability (RT) is defined as the ability to describe and follow the life of a requirement. RT helps developers ensure that relevant requirements are implemented and that the source code is consistent with its requirement with respect to a set of traceability links called *trace links*. Previous work leverages Parts Of Speech (POS) tagging of software artifacts to recover trace links among them. These studies work on the premise that discarding one or more POS tags results in an improved accuracy of Information Retrieval (IR) techniques.

Objective: First, we show empirically that excluding one or more POS tags could negatively impact the accuracy of existing IR-based traceability approaches, namely the Vector Space Model (VSM) and the Jensen Shannon Model (JSM). Second, we propose a method that improves the accuracy of IR-based traceability approaches.

Method: We developed an approach, called *ConPOS*, to recover *trace links* using constraint-based pruning. *ConPOS* uses major POS categories and applies constraints to the recovered trace links for pruning as a filtering process to significantly improve the effectiveness of IR-based techniques. We conducted an experiment to provide evidence that removing POSs does not improve the accuracy of IR techniques. Furthermore, we conducted two empirical studies to evaluate the effectiveness of *ConPOS* in recovering trace links compared to existing peer RT approaches.

Results: The results of the first empirical study show that removing one or more POS negatively impacts the accuracy of VSM and JSM. Furthermore, the results from the other empirical studies show that *ConPOS* provides 11%-107%, 8%-64%, and 15%-170% higher precision, recall, and mean average precision (MAP) than VSM and JSM.

Conclusion: We showed that *ConPos* outperforms existing IR-based RT approaches that discard some POS tags from the input documents.

Keywords: Requirements Traceability (RT), Parts of Speech (POS), Information Retrieval (IR), Trace links

1. Introduction

Requirement traceability (RT) is defined as the ability to describe and follow the life of a requirement, in both a forward and backward direction [1]. RT thus helps developers ensure that relevant requirements are implemented and that the source code is consistent with its requirement [2, 3] with respect to a set of traceability links called *trace links* [4]. Each *trace link* describes the association/connection between a requirement and a source-code entity that implements the requirement. Intuitively, as changes are introduced to requirements documents and/or source code, it is desirable for RT to keep all trace links up to date. In practice, however, developers do not always keep trace links updated during continuous software maintenance and evolution tasks [5, 6]. Thus, trace links between requirements and source code entities tend to become obsolete. Manually creating trace links is an effort intensive task. Information retrieval (IR) techniques are often used to recover trace links semi-automatically or automatically, with various accuracies [7, 8]. IR techniques are used to compute the similarity between the requirements and the source code comments and identifiers. A high similarity means a potential trace link between a requirement and a source code entity. IR techniques generate a ranked list in descending order of potential trace links based on textual similarity.

Due to noise and/or misleading textual information [9, 10] in the requirements and the source code, IR techniques may produce many false-positive links while pushing true-positive links further down in the ranked list, resulting in low precision and recall. The poor accuracy of IR techniques potentially lead to excessive developer effort for examining the resulting trace links, which has motivated researchers to improve IR techniques. To that end, various approaches have been developed to

*Corresponding author
Email addresses: cnali@memphis.edu (Nasir Ali), hcai@eecs.wsu.edu (Haipeng Cai), abdelw@ece.concordia.ca (Abdelwahab Hamou-Lhadj), jhassine@kfupm.edu.sa (Jameleddine Hassine)

reduce noises in the input documents. For instance, a heuristic is to only index nouns of software artifacts [11, 12]. However, a requirement may not be complete without a proper verb, and removing a verb could result in the loss of important semantic information. For example, a requirement R_1 for an email client "Email client shall open and display a message in a single window" contains three verbs (two main verbs, i.e., *open* and *display*, and one model verb, i.e., *shall*) and four nouns (email, client, message, and window). In the source code, although class names are frequently constructed from nouns, verbs often appear in the identifiers of method names; one reason could be that the Java Language Specification recommends that method names should be verbs or verb phrases [13, 14]. Similar naming conventions are applicable to other languages as well (e.g., C [15]). Several studies have shown that method names are an important source of information for RT tasks [16, 9]. Therefore, failing to include verbs in the RT recovery process may result in linking the requirements to the incorrect source code snippets. For example, an IR technique may link requirement R_1 to all classes in the source code that are responsible for sending, receiving, and deleting "messages", leading to large imprecision.

To recover trace links, previous IR-based works have exploited part-of-speech (POS) tagging, a technique that marks up a word in a given text corpus as corresponding to a particular part of speech (e.g., noun, verb, adjective, etc.). Yet, existing POS taggers are not 100% accurate in tagging source code identifiers [17]. One of the reasons is that a POS tagger attempts to effectively tag terms based on the context of proper English sentences and grammatical structures [18]. However, source code identifiers are bags of words, which do not contain such context information. Removing one POS or combinations of POSs from a text corpus would also result in loss of semantic information such as word contexts that IR techniques typically utilize. Therefore, we hypothesize that excluding one or more major POS may have negative impact on the accuracy of the IR techniques in recovering trace links from the text corpus. In the rest of the paper, we use the term "all POSs" to refer to the major POS categories, i.e., nouns, verbs, adjectives, adverbs, and pronouns. We believe that POS categories such as prepositions, determiners, interjection, and conjunctions are not relevant in our context. These issues with existing RT techniques and our observations motivated us to develop a novel approach.

Thus, towards improved RT accuracy and lowered cost for inspecting the results of RT techniques, we propose a new approach to trace link recovery, called *ConPOS*. ConPOS covers two levels of requirements (i.e., pre- and post-requirements) as well as two different granularity levels of resulting trace links (i.e., class and method). A pre-requirement (*resp.* post-requirement) is defined as a requirement being in its lifetime point prior to (*resp.* after) its inclusion into the final requirements specification. Feature requests in an issue tracking system could also be considered as pre-requirements. *ConPOS* starts with tagging all POSs of a requirement (without excluding any particular POS), then uses two IR techniques, namely Vector Space Model (VSM) and Jensen-Shannon Model (JSM), to recover baseline trace links between given requirements and source code files. Next, as a critical step of our approach, false-positive trace links are identified and cut off according to a constraint-based pruning strategy. Given the fact that *verbs* play an important role in expressing actions in both requirements and source code (e.g., through method names), ConPOS opts for *verbs* as the default constraints choice for the pruning; However, other types of POS constraints can also be used. For each link, ConPOS looks for a tagged verb of a requirement in the linked source code files; if the verb appears in both the requirement and the source code then ConPOS approves the link. It is important to note that ConPOS refines the baseline trace links using the constraint as a post-processing step after the baseline trace links are produced using all POSs. This is drastically different from recovering the trace links using nouns as the only POS [11, 12].

In order to validate our approach, we have conducted an empirical study using four medium-size open-source systems, namely iTrust, Lynx, Pooka, and SIP Communicator. We have measured the effectiveness of our approach using *precision*, *recall*, and mean average precision *MAP* (among all trace links produced) of those that are true-positive links. In addition, using the same four subject software, we have evaluated the effectiveness of ConPOS compared to techniques based on the removal of specific POS types from input documents (e.g., noun-based indexing) as baselines. Our results show that ConPOS provides 11%-107%, 8%-64%, and 15%-170% higher precision, recall, and MAP, respectively, than the baselines. Hence, we have revealed *for the first time* that using all POSs is essential for effective IR-based trace link recovery, which is *drastically different from previous findings* suggest that removing POSs could bring gains in precision/recall [11, 12]. Furthermore, using the proposed ConPOS approach, we have demonstrated that applying constraints for pruning as a filtering process, after recovering all trace links using all POSs, can significantly improve the effectiveness of IR-based techniques in recovering trace links.

It is noteworthy that trace links in practice have a broader scope than what *ConPOS* addresses, including such links among requirements, design documents, test cases, and bug reports, etc. In this paper, however, we focus on improving the accuracy of trace links between requirements and source code, which is an important element of the complete trace links needed by developers. In a realistic scenario, users will use results of *ConPOS* together with the other kinds of trace links produced by other, complementary approaches.

In the paper, we make the following contributions:

- We have developed a novel approach to trace link recovery, called *ConPOS*, that introduces the concept of constraints on POS and leverages such constraints to prune false-positive trace links (as a post-processing step), hence to improve RT effectiveness.

- We have conducted an empirical study to assess the effectiveness of state-of-the-art existing IR-based trace link recovery techniques. To this end, we have used datasets covering different granularity levels of trace links, various levels of requirements, and several types of POSs and POS combinations. We have shown that, generally, the removal of any POS from input documents would reduce the effectiveness of these techniques and that the use of all POSs is important for effective trace link recovery.

- We have evaluated the effectiveness of *ConPOS* and showed its advantages over existing IR-based trace link recovery techniques. We have also studied the effects of using different types of POSs as constraints on ConPOS effectiveness, and showed that verbs are the most effective POSs for the constraint-based trace link refinement.

The rest of this paper is organized as follows. In the next section, we provide a brief description of the state-of-the-art IR techniques and use of POS in software maintenance. Section 3 describes our proposed approach *ConPOS* in detail and our implementation of *ConPOS*. Section 4 presents the empirical validation of ConPOS. Sections 5 and 6 discuss more about our study results and threats to the validity of our findings. Finally, Section 7 provides concluding remarks of this work.

2. Related work

In the last couple of decades, automated RT has gained researchers' attention. IR techniques have shown promising results for automated RT [19, 20]. Borg et al. [20] conducted a Systematic Mapping (SM) study on IR-based trace recovery that contains 79 publications along with their corresponding empirical data. The authors [20] found that the majority of the studies applied algebraic IR models [7, 21, 22, 12], e.g., Vector Space Model (VSM) and Latent Semantic Indexing (LSI), to recover trace links. Comparisons have been made among different IR techniques, e.g., [23] and [19], with inconclusive results.

Borg and Runeson [24] have synthesized results from the published empirical data corresponding to 25 studies identified in [20]. The authors [24] observed that many comparing studies found that VSM presented the best results. In addition, four studies presented in three publications [19, 23, 25] found that measuring similarities using Jensen-Shannon divergence [26] performed trace recovery with a similar accuracy to VSM. The results of the study [24] suggest that the classic VSM performs better or as good as other models. Hence, VSM and JSM perform favorably in comparison to more complex techniques, such as LSI [21] or latent dirichlet allocation [27]. These findings are also in line with the claim by Falessi et al. [28] and Ali et al. [29], that simple IR techniques are typically the most useful. Yet, algebraic models, e.g., VSM [7] and probabilistic model, e.g., the JSM model [19], are a reference baseline for both feature location [22, 30] and RT [7, 31].

IR techniques depend on textual information of documents to recover trace links. Due to the noise in the textual information, e.g., misleading textual information [9], IR techniques could recover false positive links and exhibit poor accuracy [11]. To improve the accuracy of IR techniques, some researchers have explored the source code structure to combine it with textual information as an extra source of information [32, 33, 34]. Antoniol et al. [32, 33] proposed an approach for automatically recovering trace links between object-oriented design models and source code. The authors used class attributes as traceability anchors to recover trace links. McMillan et al. [34] proposed a technique for indirectly recovering trace links by combining textual information with structural information. The authors conjectured that related requirements share related source code elements. Grechnik et al. [35] proposed an approach for automating part of the process of recovering trace links between types and variables in Java programs and elements of use-case diagrams. Diaz et al. [36] leveraged source code ownership information to capture relationships between source code artifacts for improving the recovery of trace links between documentation and source code.

The large number of false positive trace links and poor accuracy of IR techniques have persuaded researchers to propose approaches to remove noise from the textual information, e.g., removing common terms [37], splitting identifiers [38], and indexing only nouns [11, 39]. Morphological analysis, e.g., stemming, is used to bring back the terms to their base or root form [37]. For example, *email* and *emails* would be converted to *email*, and an IR technique would treat them as one term. Inverse document frequency (IDF) penalizes common terms to reduce the noise. For example, if a term appears in all the documents then IDF will assign less weight to the term [7].

The importance of a term position in different textual documents have been explored by various researchers [40, 41, 42]. They observed that if a term appears in different zones of a document, then its importance changes. Yet, the state-of-the-art TF/IDF is the most commonly used term weighting scheme [20]. Dit et al. [38] used various identifier splitting algorithms to measure their impact on IR-based feature location. Their results show that only splitting identifiers manually achieves slightly better accuracy.

Some researchers have explored POS tagger to analyze the textual documents [11, 43, 44]. The POS tagger tags all the terms in the corpora according to the context and grammar of a sentence. It is successfully used in various software maintenance tasks [11, 44, 45]. Etzkom et al. [44] used a POS tagger to tag source code entities, i.e., attributes and method names. They used tagged source code entities to generate meaningful summary of the source code modules. Zou et al. [43] incorporated the use of phrases detection and construction from requirements using POS tagger to dynamically trace requirements. Their approach depends on a project glossary to find additional phrases and weight the contribution of key phrases and terms. Abebe and Tonella [45] used POS tagger to extracts concepts from the source code identifiers. They parsed the source code to build a dependency tree and extracted an ontology by mapping linguistic entities. The built ontology was then used to improve the accuracy of concept location. Shokripour et al. [46] extracted only nouns from the information sources, i.e., repository commits, identifiers and comments in the source code, and information from previously fixed bugs, to develop a bug report assignment recommendation system. Capobianco et al. [47, 11] proposed an heuristic to discard all POSs except nouns for a traceability task. They conjectured that keeping only nouns could improve the accuracy of IR techniques. In some cases, their approach produced better and in some cases worse results than a baseline IR technique. They observed that some incorrect POS tagging caused poor accuracy.

3. ConPOS: Combining POS with IR Techniques

In this section, we describe our proposed *ConPOS* approach to trace link recovery. Unlike other approaches in the literature, *ConPOS* does not exclude any POS from the input documents. However, what sets *ConPOS* from existing techniques is that it leverages POS information of requirements documents only to improve the effectiveness in trace link recovery. That is, *ConPOS* does not perform any POS-tagging tasks with the source code.

Specifically, *ConPOS* follows five key steps to recover trace links:

1. It takes as input documents both the requirements and source code, and performs POS-tagging only in the requirements document.
2. It obtains the baseline trace links from existing approaches (we refer to as baseline approaches) with both input documents.
3. For each produced baseline trace link, *ConPOS* looks for a tagged verb of a requirement in the linked source code files.
4. If a verb of the requirement is present in the linked source code files then *ConPOS* approves the link; otherwise, the link is pruned from the baseline result.
5. It outputs the remaining trace links.

Figure 1 shows the high-level architecture of ConPOS, where the major steps are detailed in the following sections.

3.1. Requirements and Source Code Preprocessing

In order to recover trace links with IR-based approaches, we performed a few standard document pre-processing steps. We used a POS tagger to tag all terms in the corresponding input documents (requirements document and source code). After POS tagging, we did two more steps: removing stop words and stemming [7, 21].

The specific workflow we followed includes five steps:

1. We used CamelCase and under_score algorithms to split source code identifiers into terms [48].
2. We removed non-alphabetical characters (e.g., numbers, special characters, etc.).
3. We converted all uppercase letters into lowercase, removed punctuation, and removed stop words (e.g., "is", "the", etc.).
4. We used POSSE[1] [17] POS tagger to tag all the source code identifiers. POSSE [17] is the existing best-performing source code identifier POS tagger compared to peer taggers according to the authors (although in absolute terms it is still not quite accurate for tagging source code). Requirements and source code comments are written in formal English. Thus, we used Standford POS tagger [49] to tag requirements and source code comments. We used tagged source code identifiers only to perform analysis on effect of removing any POS from the whole corpora. For example, in the case of comparison with noun-based trace link recovery approach we used tagged source code identifiers to select only nouns.
5. We performed stemming on the remaining terms in the input documents using the Porter [50] English stemmer, which removes the postfix of an English term and brings the term to its root (e.g., "creates" → "create").

3.2. Constraint-based Pruning

At the core of ConPOS is the process of pruning baseline trace links using a filter (referred to as *constraint*). To facilitate the description of this constraint-based pruning step, we use a set of notations as described in Table 1. Note that $\alpha(R_n, C_s, k_j) = \left|\theta(R_n, k_j) \cap C_s\right|$.

In attempting to prune false positives from the baseline trace links, for a trace link from a requirement R_n to associated source code entity C_s, ConPOS calculates the similarity score of the link with respect to constraint k_j (denoted as $\phi(R_n, C_s, k_j)$) as follows. If $\alpha(R_n, C_s, k_j) \geq 1$, the score is $\sigma(R_n, C_s) * 1.\lambda$ where λ is the number of occurrences of POS of the constraint type (i.e., k_j) in C_s; otherwise, the score is 0. By assigning a zero score to a link, ConPOS essentially removes that link, while using λ as a reward is to push upwards trace links that have more matches between the requirement and the source code entity with respect to the constraint. By default, ConPOS uses *verbs as constraints* for false-positive trace link pruning.

For example, if a requirement R_1 is linked to source code entity C_1 with the similarity score 0.4, and R_1 contains 2, i.e., $\lambda = 2$, verbs that are available in source code, then the new similarity score of the trace link would be 0.48. The higher similarity score would push link higher in the ranked list. If R_1 and C_1 don't share any verb then ConPOS will discard the trace link.

3.3. IR Engine

The IR engine in *ConPOS* takes as input both requirements documents and source code documents, and produces baseline *trace links*, as well as the similarity function $\alpha(R_n, C_s, k_j)$ to prune false-positive links based on the constraint. The IR engine builds a $m \times n$ term-by-document matrix, where m is the number of all unique terms that occur in the documents and n is the number of documents in the corpus. Then, each cell of the matrix contains a value $w_{i,j}$, which represents the weight of the i^{th} term in the j^{th} document. A weight represents the importance of a term in the corpus of all terms. Various term weighting schemes are available to compute the weight of a term [19, 7]. In this paper, we use the TF/IDF weighting scheme.

IR techniques compute the similarity between two documents based on the similarity of their terms and/or the distributions thereof. A higher similarity value between two documents

[1] https://github.com/samirgupta/POSSE/

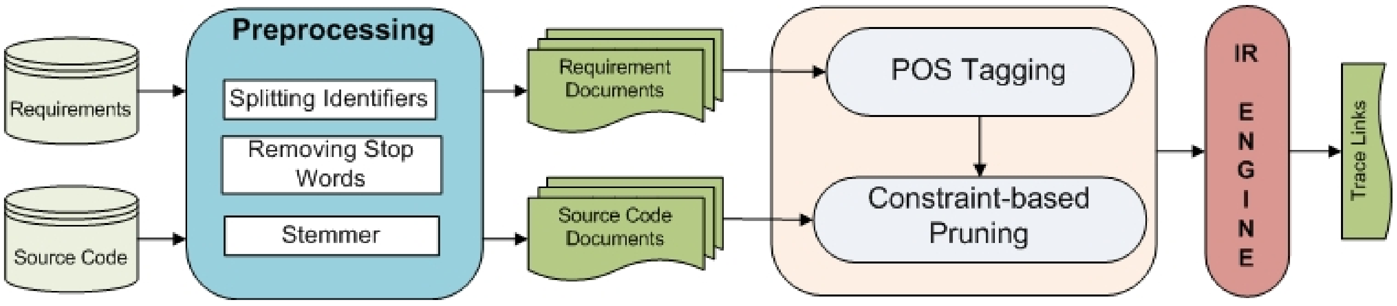


Figure 1: Overview of the proposed ConPOS approach to trace link recovery

Table 1: Summary of the Notations Used in ConPOS Description.

Notation	Meaning
R_n	A particular requirement
R	A set of requirements
C	A set of source code Entities
p_i	A term in a requirement
$P(R_n)$	The set of all terms consisting of POS in a requirement R_n
k_j	A particular type of POS
$K(R_n, k_j)$	The set of POS of type k_j in a requirement R_n
$\beta(p_i)$	Returns the type of a term p_i
$\theta(R_n, k_j)$	Returns the set $K(R_n, k_j)$ of POS of type k_j from a given requirement R_n
$\sigma(R_n, C_s)$	Returns the similarity between R_n and C_s
$\alpha(R_n, C_s, k_j)$	Returns the total number of terms of type k_j in a requirement R_n that appear in a source code entity C_s

suggests a stronger potential trace link between them. Thus, our proposed approach does not depend on a particular IR technique: various IR techniques [19, 7, 21] can be incorporated in our approach to serve the IR engine using the similarity function of ConPOS. The IR technique computes the baseline trace links for ConPOS with *all* POSs utilized and then the similarity function with the constraint used for filtering prunes suspicious false positives from the baseline links.

4. Empirical Validation of ConPOS

Existing studies have reported that nouns and verbs are better discriminators of a document than other kinds of POS [51, 39]. For example, Capobianco et al. [11] showed that considering nouns only could improve the accuracy of IR-based RT approaches. We have reexamined empirically such findings while verifying our hypothesis that excluding some POSs from software artifacts could negatively impact the accuracy of IR-based trace link recovery techniques (see research question RQ1 and its corresponding Hypothesis 1 (Sect. 4.2)). Furthermore, we have conducted an empirical study to evaluate the effectiveness of *ConPOS* in recovering trace links compared to existing peer RT approaches (see research question RQ2 and its corresponding Hypothesis 2 (Sect. 4.2)). Finally, to gauge and understand the improvement of *ConPOS* over existing peer approaches, we compared the effectiveness of *ConPOS* with existing IR-based and Nn-based RT approaches—as per the scope and focus of this work, we consider (and compare our work with) only approaches that use POS as a major means to recover *trace links*. The improvement is expected to bring reduction of the effort of software maintenance practitioners in tracing requirements and validating trace links to source code entities (see RQ3 and its corresponding Hypothesis 3 (Sect. 4.2)).

The main goals of our empirical study can be stated as follows:

- **Goal 1**: Provide sufficient empirical evidence that removing any POS does not improve the accuracy of IR techniques (Our proposed *ConPOS* considers all POSs). This goal is formulated using the following research question:
  RQ1: Does removing any major POS improve the accuracy of IR techniques?

- **Goal 2**: Provide sufficient empirical evidence that the proposed *ConPOS* approach would lead to better accuracy compared to IR-based RT approaches. This goal is formulated using the following research question:
  RQ2: Does ConPOS lead to better accuracy than IR-based RT approaches?

- **Goal 3**: Provide sufficient empirical evidence that the proposed *ConPOS* approach produces a better accuracy of the trace links compared to noun-based indexing approaches. This goal is formulated using the following research question:
  RQ3: How does the accuracy of the trace links recov-

ered by ConPOS compare with a noun-based indexing approach?

4.1. Study Setup

In this section, we describe our empirical validation following the templates and recommendations presented in Wohlin et al. [52], Juristo and Moreno [53], Kitchenham et al. [54], and Jedlitschka and Ciolkowski [55].

4.1.1. Software Subjects

We have chosen four subject software systems for our studies, as described in Table 2. For each subject, the table gives the major characteristics relevant to our experiments wherever available, including version, number of source lines of code (*SLOC*), the primary programming language in which the subject is written (*PL*), number of requirements (*#Reqs*), number of classes (*#Classes*), number of methods (*#Methods*), and the number of true-positive (ground-truth) trace links (*#TP Links*). In selecting these subjects, we used the following three criteria. First, we chose open-source systems to facilitate the reproduction of our experiments by other researchers. Second, we avoided small systems that are not representative of most systems handled by developers in practice. The four chosen systems are all medium-sized systems—we gauged the size of software in terms of its number of source lines of code (SLOC), and regarded over 10,000 SLOC as a medium size. We did not choose very-large ones because using large systems will make it hard to manually recover and validate their trace links for ground-truth construction. Third, we chose subjects such that different granularity levels of trace links are covered. For example, the Lynx subject contains method-level trace links (i.e., links from requirements to methods) while all the other three subjects contain class-level trace links (i.e., links from requirements to classes).

iTrust[2] is an application that provides patients with a means to keep up with their medical history and records as well as communicate with their doctors. Lynx[3] is a basic textual web browser entirely written in C. Pooka[4] is an e-mail client written in Java using the JavaMail API. SIP[5] is an audio/video Internet phone and instant messenger. These subjects covered both pre- and post-requirements. Specifically, Pooka, SIP, and Lynx contain pre-requirement and iTrust contains post-requirements. In comparison, pre-requirements contain less and more vague textual information [12], whereas post-requirements are more detailed.

True-positive (ground-truth) trace links: We manually created the ground-truth trace links for these subjects. Pooka and SIP datasets were not specifically created for this study. Both of the datasets are used in other studies as well [56, 57]. For the sake of completeness, we briefly explain the process of building trace links for Pooka and SIP. Three Ph.D. students recovered trace links between the requirements and associated source code of each subject. All the students have more than 3 years of Java programming experience and similar experience in the C programming language. The students read the requirements and manually searched for source code snippets that implemented corresponding requirements. They used Eclipse source code search feature to perform source code searches. For example, if a requirement is related to “send an email” then the students used such search queries as “send email”, “send mail”, and “email send” to locate source code snippets that implement the “send an email” feature. The students stored all manually-built trace links in a CSV file. Each CSV file contains source and target document names. Then, three software engineering professors, who have deep knowledge on software requirements and RT along with rich experience in Java and C, followed the same process as the students to link requirements to the source code. At the end, a majority-voting mechanism was used to accept or reject such trace links—a link was accepted if two or three professors agreed on that link being correct. We did not use, at any point of the process, any automated technique to build the oracles (ground truth). For iTrust, we used the traceability oracle provided by the original developers of this software. For Lynx, we used the traceability oracle provided by [58]. To facilitate replication and reuse, all our experimental datasets including the ground-truth trace links are publicly available[6].

4.1.2. Baseline Approaches and Baseline trace links

We have selected two baseline approaches, JSM and VSM, and took their resulting trace links as the baseline RT results. The selection of these two techniques as baseline approaches was motivated by the facts that (1) they were shown to outperform peer techniques according to previous comparative studies on the effectiveness of IR-based traceability approaches [19, 23], and (2) they do not depend on any parametric tuning, hence potentially reducing internal validity threats to our study results. Table 3 shows the used baseline approaches as well as their corresponding POS-based approaches, where the approach subscript denotes the type of used POS, e.g., Adj, Nn, Vb, Adj+Nn, Adj+Vb, or Nn+Vb. For example, VSM_{Adj+Vb} denotes the VSM approach using adjectives and verbs as POS categories.

To recover baseline trace links, we followed the same steps as described in [7] and [19].

4.2. Hypotheses

The experiments were planned with the purpose of testing the following hypotheses:

1. **Hypothesis 1**: To answer RQ1, we use all major POSs and their combinations to recover trace links. Table 4 shows the selected POS sub-categories, divided into three main classes: Adj (adjective), Nn (noun), and Vb (verb). We compare the average precision (AP) of POS-based (e.g.,

[2]`http://agile.csc.ncsu.edu/iTrust/`
[3]http://lynx.isc.org/
[4]`http://www.suberic.net/pooka/`
[5]`http://www.jitsi.org`

[6]http://factrace.net/nasir/emse/

Table 2: Characteristics of Software Systems Used as Experimental Subjects

Subject	Version	SLOC	PL	#Reqs	#Classes	#Methods	#TP Links
iTrust	10	19,604	Java	35	218	–	186
Pooka	2.0	244,870	Java	90	298	–	507
Lynx	2.8.5	–	C	128	–	2,067	376
SIP	1.0	486,966	Java	82	1,771	–	871

Table 3: Baseline approaches and corresponding POS-based approaches

Baseline approach	Corresponding POS-based approach
VSM	VSM_{Adj}, VSM_{Nn}, VSM_{Vb}, VSM_{Adj+Nn}, VSM_{Adj+Vb}, VSM_{Nn+Vb}
JSM	JSM_{Adj}, JSM_{Nn}, JSM_{Vb}, JSM_{Adj+Nn}, JSM_{Adj+Vb}, JSM_{Nn+Vb}

Table 4: Main and Sub Categories of POS

	Main POS	Sub Categories of POS
1	Adjectives (Adj)	Comparative adjective
		Superlative adjective
2	Nouns (Nn)	Singular noun
		Plural noun
		Proper singular noun
		Proper plural noun
		Personal pronoun
		Possessive pronoun
3	Verbs (Vb)	Adverb
		Comparative adverb
		Superlative Adverb
		Verb, past tense
		Verb, gerund/present participle
		Verb, past participle
		Verb, non 3rd ps. sing. present

noun-based [11]) RT approaches with the two baseline (IR-based) approaches. Our null hypothesis is as follows:

H_0- JSM-JSM$_{POS}$: *There is no statistical difference in the average preicison of the recovered trace links between JSM and JSM$_{POS}$ approaches.*

where *POS* could be Adj, Nn, Vb, Adj+Nn, Adj+Vb, or Nn+Vb. Similarly, we have six null hypotheses for VSM also (i.e., **H_0- VSM-VSM$_{POS}$**).

2. **Hypothesis 2**: To answer RQ2, we use our proposed approach ConPOS to recover trace links and compare the average precision with baseline IR-based RT approaches JSM and VSM. For RQ2, we formulate the following null hypothesis:

H_{01JSM}: *There is no statistical difference in the average precision of the recovered trace links between JSM and ConPOS$_{JSM}$.*

We have similar null hypotheses H_{01VSM} for VSM.

3. **Hypothesis 3**: To answer RQ3, we use our proposed approach ConPOS to recover trace links and compare the average precision with Nn-based RT approach [11]. For RQ3, we formulate the following null hypothesis:

$H_{01JSM_{Nn}}$: *There is no statistical difference in the average precision of the recovered trace links between JSM$_{Nn}$ and ConPOS$_{JSM}$.*

We have similar null hypotheses for **$H_{01VSM_{Nn}}$**.

4.3. Analysis Approach

In this section, we briefly present the used accuracy metrics and discuss the significance of the statistical analysis.

4.3.1. Accuracy Metrics

In our studies, we use four standard metrics for evaluating the effectiveness of an IR-based RT approach, as defined below.

Precision is defined as the ratio of the number of relevant trace links retrieved to the total number of retrieved links by an approach. If precision is 1 (or 100%) then all the recovered trace links are relevant ones (i.e., true positives).

$$Precision = \frac{|\{relevant\ links\} \cap \{retrieved\ links\}|}{|\{retrieved\ links\}|}$$

Recall is defined as the ratio of the number of relevant trace links retrieved to the total number of relevant trace links. If recall is 1 (or 100%), then all relevant (true-positive) trace links have been retrieved by an approach.

$$Recall = \frac{|\{relevant\ links\} \cap \{retrieved\ links\}|}{|\{relevant\ links\}|}$$

Average Precision: While precision and recall together capture the accuracy of an RT approach, high accuracy does not imply that all true-positive links would be placed at the top of the resulting list of trace links. In fact, it is quite possible that a RT approach provides better accuracy than other approaches yet all the true-positive links are down in that list, with which a developer would have to manually discard false-positive links before reaching a true-positive link. Thus, beyond high accuracy, it also is important to bring the true-positive links up in the results in order to save developers' effort. Therefore, to measure trace link recovery effectiveness, we also consider the average precision (AP) [59, 60] of true-positive trace links in the result of a RT approach. We considered all the candidate links retrieved for a requirement.

$$AP = \frac{1}{|R|} \bullet \sum_{i=1}^{n} (Prec(i) \bullet relevance(i))$$

Where:

- $|R|$: total number of relevant documents
- $Prec(i)$: precision at top i documents
- $relevance(i)$: relevance of the document at rank i, equaling 1 if the document is relevant, 0 otherwise

Mean Average Precision: To compute mean average precision (MAP) [59, 60], we consider a requirement as a topic/query and retrieved source code snippets as documents. Thus, MAP for a set of queries is the mean of the average precision scores for each query.

$$MAP = \frac{AP}{Q}$$

Where Q is the number of queries.

4.3.2. Statistical Analysis

It is quite possible that an approach has better MAP than the others, but is the approach statistically significant? To make sure, improvement in MAP is statistically significant, we perform statistical analysis on AP of each requirement. To evaluate the performance of each requirement's trace, we select AP metric to accept or reject our null hypothesis. We use each AP value as a datapoint in our statistical test to measure the significant difference of accuracy.

For that purpose, we needed to perform appropriate statistical tests with respect to the distribution of the datapoints (i.e., the metric values). We first used a Shapiro-Wilk test to analyze the distribution and found that our data set does not follow a normal distribution. Thus, we chose a non-parametric test (Mann-Whitney test in this work), which does not require the underlying data set to be normally distributed, for our hypothesis testing. We use the $\alpha = 0.05$ to accept or refute null hypotheses. We test multiple hypotheses, therefore the likelihood of incorrectly rejecting a null hypothesis (i.e., making Type-I error) increases. Therefore, we perform Bonferroni, (i.e., m/α where, m is total number of null hypothesis), correction on α. We apply Bonferroni correction by dividing $0.05/6$, $0.05/1$, and $0.05/1$ to adjust α value for Hypothesis-2, and Hypothesis-3 respectively.

4.4. Variables

We use AP as the dependent variable. As per the independent variables, we use all RT approaches (listed in Table 3) to verify our first hypothesis and only JSM, VSM, JSM$_{Nn}$, VSM$_{Nn}$, ConPOS$_{JSM}$, and ConPOS$_{VSM}$ to verify Hypotheses 2 and 3.

4.5. Experimental Procedure

To verify our first hypothesis (i.e., Hypothesis 1), we proceeded as follows (see Fig. 2). For each subject, to obtain trace links from a POS-based approach X_{POS}, we first tagged all terms in the two input documents (requirements and source code) using POSSE [17]. Then, we divided each tagged document D into six segmental documents: D_{Adj}, D_{Nn}, D_{Vb}, D_{Adj+Nn}, D_{Adj+Vb}, and D_{Nn+Vb}, which contains only adjectives, nouns, verbs, adjectives and nouns, adjectives and verbs, and nouns and verbs, of the original document D, respectively. For example, to generate D_{Nn} from D, we removed all terms other than nouns from D—as a result, the content of D_{Nn} may not consist of complete/readable sentences. Next, we used the baseline approach (X, which is JSM or VSM) associated with the POS-based approach (X_{POS}) to recover trace links from the two segmental documents. As such, we performed seven trace link recovery tasks per baseline approach and subject: one task with the baseline approach itself using the two original input documents, and six tasks each with one of the six POS-based approaches corresponding to the baseline approach using the two segmental documents derived from those two original documents. As mentioned earlier, in each of these tasks, we generated t (where t is total number of requirements) sets of trace links.

To verify our second and third hypotheses (i.e., Hypothesis 2 and Hypothesis 3) and in addition to precision, recall, and MAP metrics, we have also computed the improvement percentage[7] as an intuitive indicator for improvement. We also adopted the same statistical analyses as in the exploratory study to measure the significance of such improvements. We have only considered three baseline approaches: (i) VSM, representative of the algebraic family of RT techniques, (ii) JSM, representative of the probabilistic family of RT techniques, and (iii) the Nn-based indexing approach to trace link recovery[11], representative of POS-based RT techniques. We denote as ConPOS$_{VSM}$ and ConPOS$_{JSM}$ the variants of ConPOS using VSM and JSM as the IR technique that produces the baseline trace links (before the constraint-based pruning), respectively. Contrary to our first experiment, we did not perform POS tagging in source code because *ConPOS* does not need/use POS information from source code. As mentioned earlier, *ConPOS* only performs POS tagging in the requirements documents. We adopted the default constraint in *ConPOS* (i.e., using verbs as the constraint). For each of the four studied subjects, we gave the input requirements and source code of the subject to *ConPOS* as its inputs. *ConPOS* generates the trace links by first computing the baseline lines using the underlying baseline approach (VSM or JSM) and then applies the constraint-based pruning with the similarity function, as described in Section 3. ConPOS discards a baseline trace link if the similarity function returns 0 with that link. To recover trace links using Nn-based indexing, we follow the same steps as Capobianco et al. [11]. For each requirements document and source code file, we tagged the terms in the input

[7]Given x (new value) and y (base value) the average percentage improvement is computed as: $\frac{(x-y)}{y}\%$

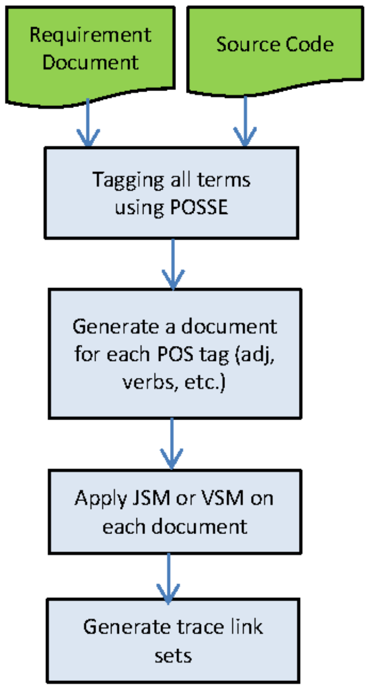

Requirement Document
Source Code
Tagging all terms using POSSE
Generate a document for each POS tag (adj, verbs, etc.)
Apply JSM or VSM on each document
Generate trace link sets

Figure 2: Procedure for verifying Hypothesis 1

text using the POSSE tagger and only kept the terms that are tagged as nouns. Finally, the baseline trace links from JSM and VSM were drawn from our first experiment.

4.6. *Experimental Results*

This section presents the results of our experimental studies as guided by the three research questions defined earlier.

4.6.1. ***RQ1: Does removing any POS improve the accuracy of IR techniques?***

The effectiveness results of the baseline IR-based approaches and corresponding POS-based approaches are listed in Table 7. Moreover, Figure 5 depicts the effectiveness results for all the threshold points we considered for a deeper analysis. Figure 6 provides an overview of an AP.

As shown in Table 7, in 83% (120/144) of the cases, using a single POS or combination of two of the POS provides significantly lower accuracy than a baseline RT approach (VSM or JSM). For AP metric, only 12.5% of the times results were significant. Thus, we cannot reject our null hypotheses H_0- JSM-JSM$_{POS}$ and H_0- VSM-VSM$_{POS}$ for all the comparisons. These results suggest that all the POS are important to recover RT trace links. However, some POS are more important than others. Results show that nouns and verbs are better discriminators of a document than other kinds of POS [51, 39]. However, excluding adjectives could cause loss of semantic information and negatively impacts the accuracy of IR-based RT approaches. For example, in the case of JSM$_{Nn+Vb}$ and VSM$_{Nn+Vb}$, 62.5% (5/8), 62.5% (5/8), and 50% (4/8) of the cases using only Nn+Vb improved precision, recall, and MAP, respectively. In the case of iTrust and Lynx, excluding any POS negatively affected the precision and the AP. Only in some of the cases for iTrust and Lynx, recall was improved. Next, we discuss the effect of each POS on IR-based RT-link recovery in detail.

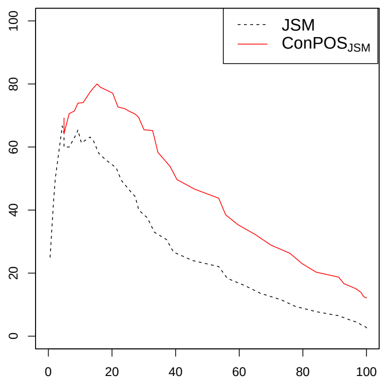
Adjectives only:. Table 7 shows that in all the cases using only Adjs provides lower accuracy than the baseline IR-based RT approaches. Adj provides extra information about a noun and/or a verb. Thus, removing nouns and verbs left the input text with incomplete information. Consequently, incomplete textual information led to lower accuracy. Figure 5 and Figure 6 show that keeping only adjectives in the source code documents provides the lowest accuracy compared to baseline RT approaches and other POS-based approaches.

Nouns only:. We also compare baseline RT approaches with the heuristics proposed by Capobianco et al. [10, 11]. Capobianco et al. [11] only considered Nns to remove the noise from the data. Their results show that in some cases (e.g., traceability from UML interaction diagrams to source code classes), Nn-based indexing did not provide any better results than an IR-based technique. The results of our exploratory study show that Nn-based indexing, in 21% (5/24, in the table for the rows JSM$_{Nn}$ and VSM$_{Nn}$) of the cases, improve the accuracy of IR-based RT approaches. The results show that 79% (19/24) of the cases Nn-based indexing provides lower accuracy than baseline RT approaches.

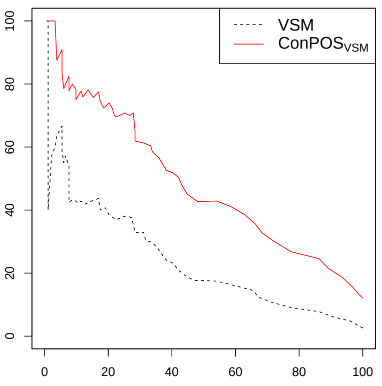
Additionally, in all the cases, by averaging the precision of POS-based approaches was always lower than that of the corresponding baseline IR-based approach. However, in 73% cases, using only Nns provides slightly better recall and MAP than the baseline technique. Figure 5 shows that the Nn-based RT approach provides better results only at some threshold points compared to the baseline approach. Figure 6 shows that Nn-based RT approach has similar AP as VSM.

Verbs only:. In the case of using only verbs, we observed a drastic decrease in both precision, recall, MAP values, in comparison to the corresponding baseline. We examined the tagged verbs of source code documents and requirements and found that due to the lack of context many of the verbs in source code documents were tagged as adjectives or nouns (e.g., “access”, “open”, and “show”). Thus, removing adjectives and nouns caused loss of semantic information, hence poor accuracy. This observation supports our conjecture that without proper contextual and grammatical information, a POS tagger could incorrectly tag source code terms. The worst-case results were seen with Lynx when using verbs only. One reason could be that the selected POS-tagger has lower accuracy on C source code, as shown by Gupta et al. [17]. In addition, Lynx has method bodies used as source-code documentation, with which there is little textual information available and very few terms were tagged as verbs.

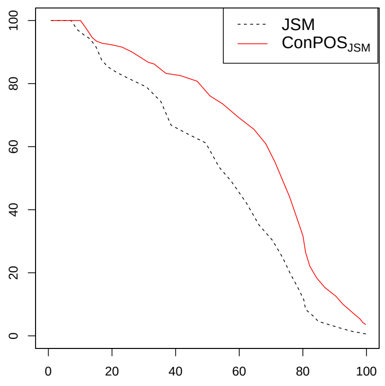
POS combinations:. In 58% (14/24) of the cases, using nouns+verbs provides better results than the baseline RT approach. Only in the case of Pooka, for all accuracy matrices, removing a POS (adj) provides slightly better results than the baseline IR techniques. (For brevity, we discuss in detail the



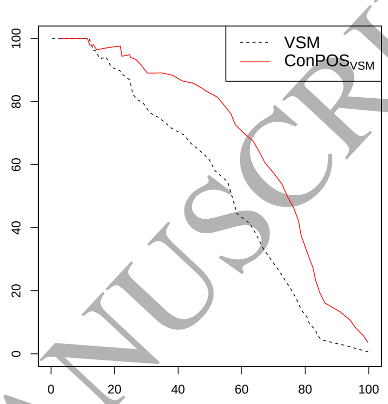
(a) iTrust



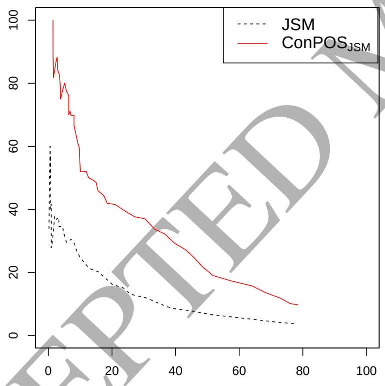
(b) iTrust



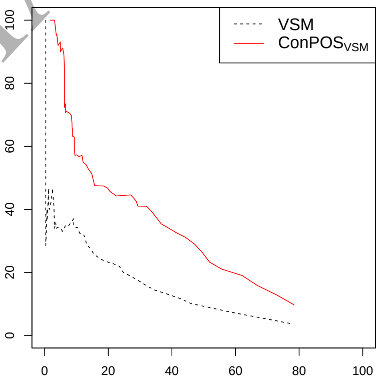
(c) Lynx



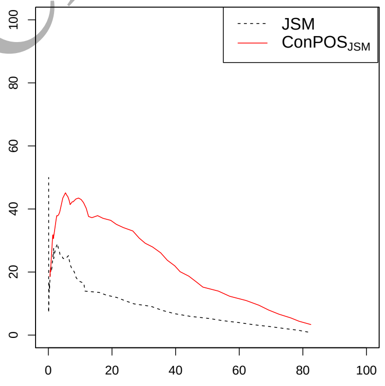
(d) Lynx



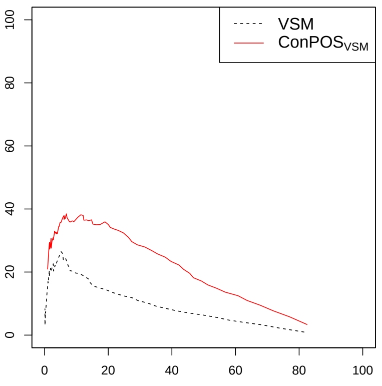
(e) Pooka



(f) Pooka

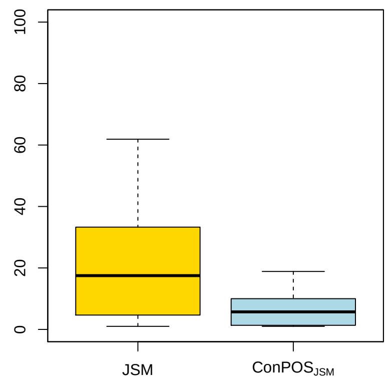


(g) SIP

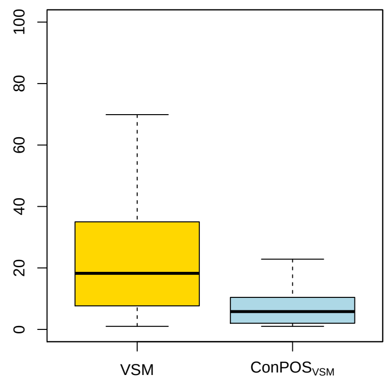


(h) SIP

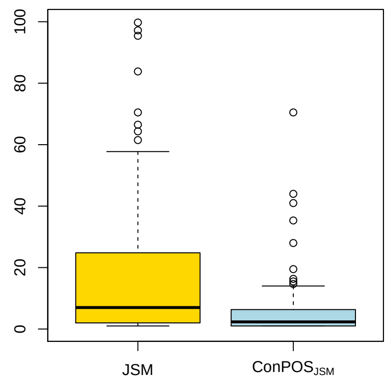
Figure 3: Precision (y axis) and recall (x axis) of ConPOS$_{JSM}$ versus JSM and ConPOS$_{VSM}$ versus VSM on the four subjects, with the similarity threshold t varying from 0.01 to 1 by step of 0.01.



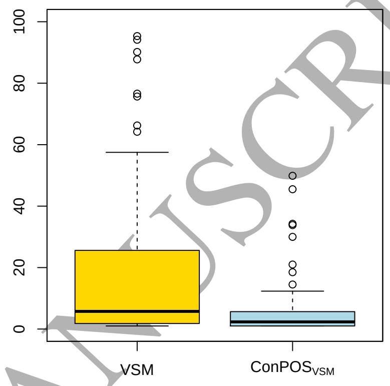
(a) iTrust



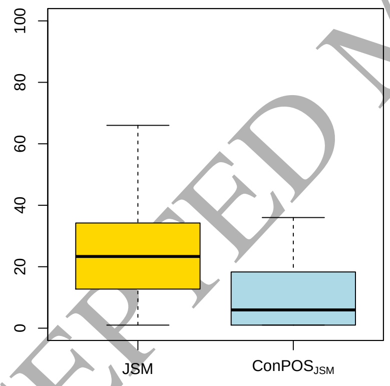
(b) iTrust



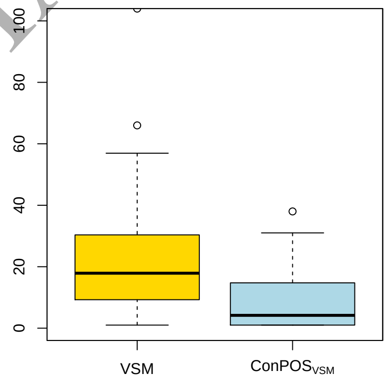
(c) Lynx



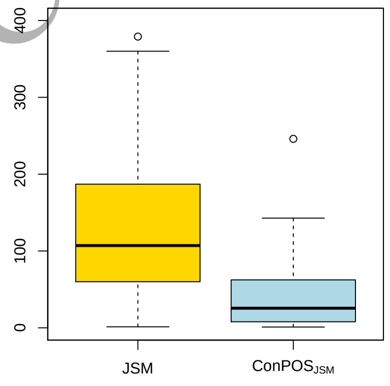
(d) Lynx



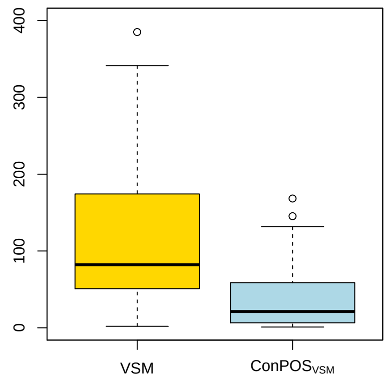
(e) Pooka



(f) Pooka



(g) SIP



(h) SIP

Figure 4: Distribution of average precision (AP) (y axis) of ConPOS$_{JSM}$ versus JSM and ConPOS$_{VSM}$ versus VSM (x axis) on the four subjects.

results on Pooka and with one baseline VSM only; similar, detailed results on other subjects and with the other baseline JSM are all available online[6]).

Figure 5 and 6 shows that using Nn+Vb provides better results on all the threshold points. In fact, Pooka is the only subject out of the four studied with which we observed this consistent contrast.

Our results show that removing any POS could cause loss of semantic information and consequently negatively affects the accuracy of IR-based RT approaches. To the best of our knowledge, this is the first study that shows that all POSs are important for an IR-based technique to recover trace links.

> Results of our exploratory study support our conjecture that excluding any POS from software artifacts could negatively affect the accuracy of IR-based trace link recovery techniques.

4.6.2. RQ2: Does ConPOS lead to better accuracy than IR-based RT approaches?

Figure 3 depicts the precision and recall of ConPOS$_{JSM}$, ConPOS$_{VSM}$ versus JSM and VSM, respectively. The charts show that *ConPOS* achieved constantly higher effectiveness than both baseline techniques. It is worth noting that the advantage of ConPOS was exhibited at any of the 100 threshold points, although the magnitude of improvement varied with different thresholds.

In the cases of Lynx and SIP, at threshold 0 *ConPOS* missed five (1.5%) and three (0.33%) true-positive links, respectively, which led to low recall at that threshold point. We further examined these bad cases and found that in all true-positive links that *ConPOS* missed the source code entities implementing the corresponding requirement do not use any verbs tagged from the requirement. In consequence, those links were discarded by *ConPOS* since it used verbs as the default constraint for pruning. On the other hand, however, for Lynx and SIP *ConPOS* was able to remove up to 82% and 70% of false-positive links, respectively, at the cost of missing only few true positive links.

Table 5 summarizes average precision and recall achieved by the two variants of ConPOS and the two baseline approaches, along with the magnitude of improvements of each variant of ConPOS over its corresponding baseline approach (as shown in the parentheses). Overall, ConPOS improved the accuracy, on average, by up to 107% in precision, 64% in recall, and 170% MAP over the baseline approaches. We further performed the statistical tests as described in Section 4.3.2 to verify whether or not the average improvements are statistically significant. Our statistical analysis results provided sufficient evidence to reject all the null hypotheses related to RQ2: in any of the comparisons, the difference between ConPOS and the compared IR-based approach was statistically significant as supported by the fact that the *p*-values of all corresponding statistical tests were well below the standard significant value (i.e., 0.05).

These results show that using verbs as constraints can greatly enhance the accuracy of an IR-based RT technique in recovering trace links.

> ConPOS provides 107%, 64%, and 170% better precision, recall, and MAP, respectively, than both base approaches VSM and JSM.

Table 5 summarizes the MAP. Figure 4 gives the distribution of the AP of all true-positive trace links. The boxplots confirmed that ConPOS always placed true-positive links at higher ranks, potentially saving the effort of practitioners in inspecting the RT results. Moreover, ConPOS lowered the rankings of true-positive links substantially compared to the two IR-based baseline approaches. Note that here the smaller the ranking numbers, the higher the ranks (i.e., the higher the true-positive links were placed in the ranked list), hence the better. For example, in the case of Lynx (VSM), on average, ConPOS$_{VSM}$ raised the rank of true-positive trace links up from 23.88 to 6.20.

> On average, ConPOS decreased the ranking of true-positive trace links by (67%-74%); thus, it places true-positive trace links substantially higher in ranked lists when compared to the baseline IR-based RT approaches.

4.6.3. RQ3: How does the accuracy of the trace links recovered by ConPOS compare with a noun-based indexing approach?

Table 6 shows average precision, recall, and MAP of ConPOS (including its variants ConPOS$_{JSM}$ and ConPOS$_{VSM}$) and Nn-based indexing approaches (VSM$_{Vn}$ and JSM$_{Vn}$). Results show that ConPOS provides 11%-107%, 8%-64%, and 15%-170% better precision, recall, and MAP, respectively, than the Nn-based indexing approach, which missed quiet a few true-positive links at 0 threshold. Additionally, putting together this table and the exploratory study results in Tables 7 reveals that ConPOS is not only more effective than Nn-based indexing approaches, but also more effective than IR-based approaches with input documents from which other types of POS (Vb and Adj) or different combinations of POS (Adj+Vb, Adj+Nn, Vb+Nn) are removed.

Further, results of our statistical analyses (same as those for RQ2) suggest rejecting all the null hypotheses related to RQ3: in any of the comparisons, the difference between ConPOS and compared Nn-based indexing approach was statistically significant as per the *p* value being far below the standard significant value 0.05.

> ConPOS provides 11%-107%, 8%-64%, and 15%-170% better precision, recall, and MAP, respectively, and decreased the rank of true-positive trace links by 67%-74%, than the Nn-based indexing approach.

In summary, as corroborated by our substantial empirical evidences, ConPOS achieves constantly better effectiveness, in terms of any of the three metrics (precision, recall, and MAP of true-positive links), than any of the baseline approaches we studied, with strong statistical significance. The key of the ConPOS approach is augmenting IR techniques with constraint-based pruning with the full use of POS information (in both

Table 5: Average precision, recall, and MAP of ConPOS versus the two IR-based baseline approaches (JSM and VSM), along with the average percentage of improvement of ConPOS over the baseline in each of these effectiveness metrics (in the parentheses). For all comparisons, p-values are below $\alpha = 0.05$.

	Precision (%)		**Recall (%)**		**MAP**	
	VSM	**ConPOS$_{VSM}$**	**VSM**	**ConPOS$_{VSM}$**	**VSM**	**ConPOS$_{VSM}$**
iTrust	48.89	67.56 (**38%+**)	23.40	27.78 (**19%+**)	0.27	0.46 (**70%+**)
Lynx	65.85	74.46 (**13%+**)	38.26	41.31 (**8%+**)	0.63	0.74 (**17%+**)
Pooka	41.31	63.96 (**55%+**)	10.70	17.56 (**64%+**)	0.23	0.47 (**104%+**)
SIP	14.14	29.29 (**107%+**)	13.06	19.16 (**47%+**)	0.12	0.29 (**142%+**)
	JSM	**ConPOS$_{JSM}$**	**JSM**	**ConPOS$_{JSM}$**	**JSM**	**ConPOS$_{JSM}$**
iTrust	34.70	49.92 (**44%+**)	40.58	46.25 (**14%+**)	0.33	0.48 (**45%+**)
Lynx	60.30	66.68 (**11%+**)	41.35	44.76 (**8%+**)	0.60	0.69 (**15%+**)
Pooka	32.24	56.67 (**76%+**)	12.69	19.94 (**57%+**)	0.17	0.41 (**141%+**)
SIP	18.45	29.43 (**60%+**)	17.07	23.75 (**39%+**)	0.10	0.27 (**170%+**)

Table 6: Precision, recall, and MAP of ConPOS versus the Nn-based indexing approach as the baseline, along with the average percentage of improvement of ConPOS over the baseline in each of these effectiveness metrics (in the parentheses).

	Precision (%)		**Recall (%)**		**MAP**	
	VSM$_{Nn}$	**ConPOS$_{VSM}$**	**VSM$_{Nn}$**	**ConPOS$_{VSM}$**	**VSM$_{Nn}$**	**ConPOS$_{VSM}$**
iTrust	36.79	67.56 (**84%+**)	24.93	27.78 (**11%+**)	0.25	0.46 (**84%+**)
Lynx	45.13	74.46 (**65%+**)	36.17	41.31 (**14%+**)	0.43	0.74 (**72%+**)
Pooka	35.17	63.96 (**82%+**)	13.02	17.56 (**35%+**)	0.19	0.47 (**147%+**)
SIP	8.40	29.29 (**249%+**)	16.51	19.16 (**16%+**)	0.10	0.29 (**190%+**)
	JSM$_{Nn}$	**ConPOS$_{JSM}$**	**JSM$_{Nn}$**	**ConPOS$_{JSM}$**	**JSM$_{Nn}$**	**ConPOS$_{JSM}$**
iTrust	27.36	49.92 (**82%+**)	42.95	46.25 (**8%+**)	0.26	0.48 (**85%+**)
Lynx	39.95	66.68 (**67%+**)	39.03	44.76 (**15%+**)	0.42	0.69 (**64%+**)
Pooka	26.05	56.67 (**118%+**)	14.75	19.94 (**35%+**)	0.16	0.41 (**156%+**)
SIP	8.15	29.43 (**261%+**)	22.54	23.75 (**5%+**)	0.09	0.27 (**200%+**)

requirements and source code for the IR technique to produce baseline trace links, and in the requirements only for the constraint-based pruning to largely remove false positives). Pruning the baseline links with a particular type of POS used as the constraint turned out to be very effective. This finding implies that the baseline approach produces a lot of false-positive links, possibly because of the inaccurate POS tagging in the source code—note that again during the constraint-based pruning, ConPOS does not tag source code but only tags the requirements, thus overcomes the limitations of existing POS taggers. Moreover, ConPOS improves in the rankings of true-positive links as well, which can be largely attributed to the use of the degree of matching between requirements and source code (i.e., λ) as a reward in the constraint-based similarity function of ConPOS. Note that the use of this reward turned out to increase the recall of ConPOS versus the baseline approaches; some of the links that would be cut off (as their similarity score fell below the threshold) were recovered by ConPOS since the reward led to higher similarity score for those links.

5. Discussion

We now provide in-depth analysis of our proposed approach and discuss observations from our empirical evaluation of ConPOS.

5.1. Discarding False-Positive Links

ConPOS not only helps to improve the accuracy of IR-based RT approaches but it also automatically removes up to 82% of false-positive links. Figure 7 shows the percentage of all removed false-positive links. For Lynx, ConPOS removes more (higher percentage of) false-positive links than for the other three subjects (especially than for Pooka and SIP).

We also observed that for Lynx ConPOS mistakenly removed five (1.5%) true-positive links (because the source code entities implementing linked requirements do not contain any verbs tagged in the requirements, as explained in Section 4.6). However, on average ConPOS rightly removed up to 82% false-positive links at the cost of only missing a few of true-positive links. Another finding is that generally the constraint-based trace link pruning (at the core of ConPOS) had almost the same effects on JSM and VSM, albeit with iTrust and SIP ConPOS$_{VSM}$ removed slightly higher percentages of false-positive trace links than ConPOS$_{JSM}$ while with the other two subjects the contrast is reversed (also with very-small differences).

Recall that the Lynx datasets contain trace links between requirements and source code at the level of methods, while the trace links are all at class level for the other subjects. Interestingly, ConPOS tends to remove more false-positive links

Table 7: Effectiveness (precision, recall, and MAP) of baseline IR-based approaches and corresponding POS-based approaches. Reported values are average of all the precision and recall computed on threshold *t* varying from 0.01 to 1 by step of 0.01. Reported MAP values are independent of any threshold. Boldface indicates a POS-based approach's result that improves over that of the corresponding IR-based baseline approach.

	Precision				**Recall**				**MAP**			
	iTrust	**Lynx**	**Pooka**	**SIP**	**iTrust**	**Lynx**	**Pooka**	**SIP**	**iTrust**	**Lynx**	**Pooka**	**SIP**
VSM	48.89	65.85	41.31	14.14	23.40	38.26	10.70	13.06	0.2679	0.6272	0.2267	0.1217
VSM$_{Adj}$	14.65	0.09	6.34	2.13	15.85	0.845	1.98	1.385	0.1268	0.0006	0.0125	0.0073
VSM$_{Nn}$	36.79	45.13	35.17	8.40	**24.93**	36.17	13.02	**16.51**	0.2499	0.4349	0.1909	0.1028
VSM$_{Vb}$	30.50	0.59	16.97	10.20	12.72	0.94	6.09	9.26	0.1481	0.0123	0.0914	0.0695
VSM$_{Adj+Nn}$	41.94	53.81	28.01	13.52	**24.37**	36.45	10.30	**16.14**	**0.2701**	0.4906	0.2025	0.1057
VSM$_{Adj+Vb}$	31.43	0.42	18.52	**22.56**	18.19	1.53	5.53	8.35	0.1978	0.0057	0.0926	0.0657
VSM$_{Nn+Vb}$	39.07	60.09	**52.68**	**15.58**	23.26	**38.87**	**14.49**	**16.45**	0.2635	0.5494	**0.2561**	**0.1282**
JSM	34.70	60.30	32.24	18.45	40.58	41.36	12.69	17.07	0.3253	0.6018	0.1656	0.1032
JSM$_{Adj}$	10.53	0.09	5.46	1.50	31.69	1.12	2.21	1.79	0.1400	0.0006	0.0138	0.0073
JSM$_{Nn}$	27.36	39.95	26.05	8.15	**42.95**	39.03	**14.75**	**22.54**	0.2600	0.4155	0.1617	0.0909
JSM$_{Vb}$	28.54	0.25	14.70	10.52	22.94	1.08	7.67	13.71	0.1708	0.0025	0.0900	0.0711
JSM$_{Adj+Nn}$	30.32	45.41	24.72	10.00	**42.59**	40.93	11.02	**21.15**	0.2944	0.4655	0.1594	0.0958
JSM$_{Adj+Vb}$	32.21	0.24	24.82	17.66	31.57	2.42	6.40	12.16	0.2334	0.0040	0.0913	0.0688
JSM$_{Nn+Vb}$	**35.19**	52.26	**41.67**	16.95	40.61	**43.77**	**16.30**	**20.84**	0.3022	0.5486	**0.2176**	**0.1368**

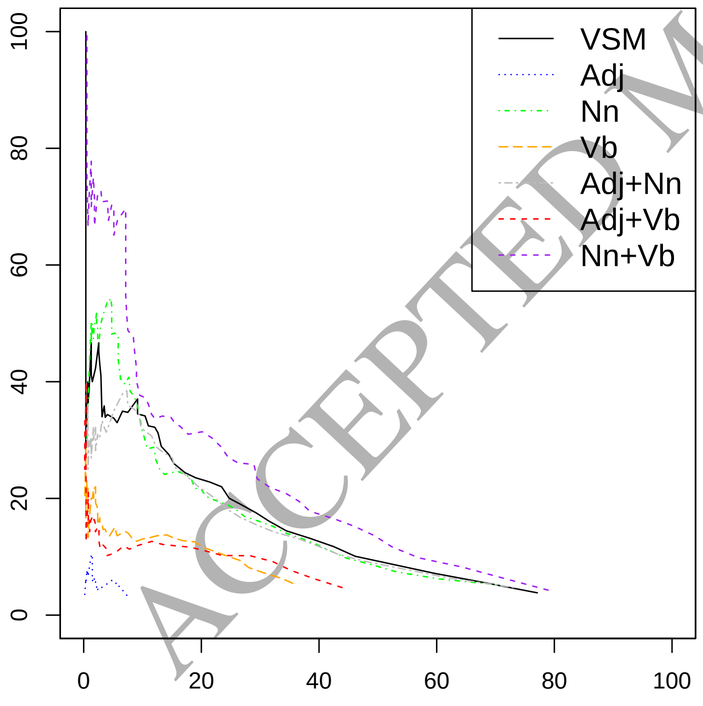


Figure 5: Precision (*y* axis) and recall (*x* axis) of trace links for Pooka produced by VSM and corresponding POS-based approaches (listed in the legend), with the threshold *t* varying from 0.01 to 1 by step of 0.01.

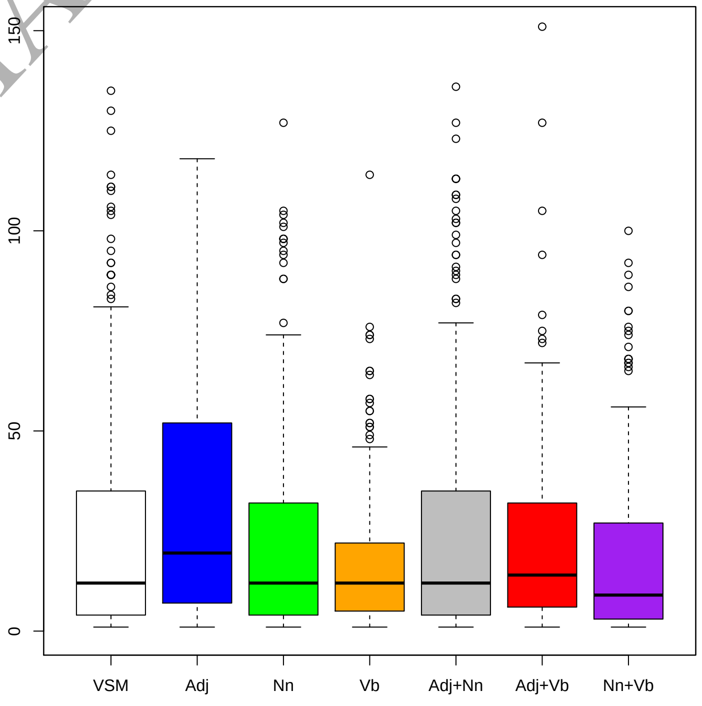


Figure 6: Distribution of average precision (AP) of true-positive links (*y* axis) among all trace links for Pooka produced by VSM and corresponding POS-based approaches (*x* axis)).

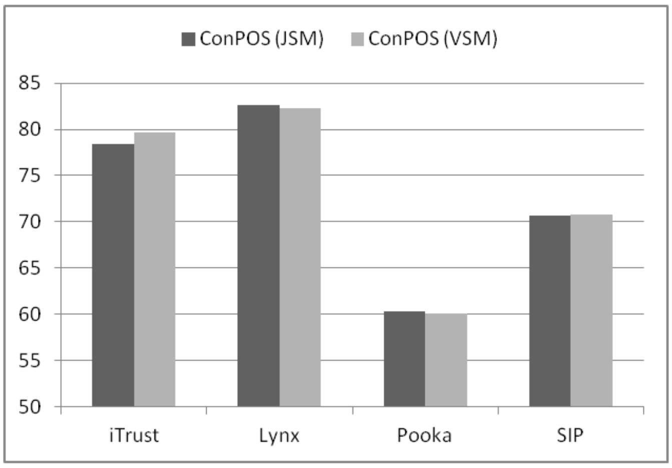


Figure 7: Percentage of false-positive trace links removed by ConPOS.

on a finer-grained granularity level than on a coarser-grained granularity level (e.g, class level). On the other hand, however, ConPOS tends to provide better accuracy on the coarser-grained granularity level than on the finer-grained granularity level. However, we cannot generalize this finding. This is because we would need more fine-grained datasets to do so, which is part of our future work (i.e., applying ConPOS to recover trace links at a larger variety of granularity levels each with a larger number of data samples).

5.2. Alternative Choices of Constraint (Nn and Adj)

While by default we use verbs as the constraint in ConPOS, ConPOS allows for any other types of POS to be used as the constraint as well. As we mentioned earlier, the justification for choosing verbs as the default constraints is that verbs define actions in requirements, thus the source code entity must implement the action of a requirement and/or an action term of a requirement must be present in the source code entity. In our evaluation study, we used the default type of constraint. Our evaluation results show that using a constraint on verbs provides better precision, recall, and MAP than VSM, JSM, and the Nn-based indexing approach. To understand the impact of the choice of constraint, we further studied two alternative choices: using Nn or Adj as the constraint in ConPOS. For example, with Nn used as the constraint, ConPOS approves an trace link if a noun (e.g., "email") that appears in a requirement is also present in the source code entity linked to that requirement; otherwise, the trace link will be pruned. We refer to this as ConPOS$_{X,y}$ the variant of ConPOS using X as the baseline IR-based technique with the type y of POS as constraint, where $X \in \{VSM, JSM\}$ and $y \in \{Nn, Vb, Adj\}$.

Table 8 shows that defining constraints on nouns led to slightly better precision 25% (2/8) of the cases, recall 75% (6/8) of the cases, and MAP 38% (3/8) of the cases, than the baseline RT approaches. Only for SIP (ConPOS$_{JSM}$), using nouns as constraint brought higher recall than the baseline and using constraint on verbs. In most of the cases, at $t = 0$, the total number of trace links recovered by ConPOS is almost the same as the baseline RT approaches, regardless of which types of POS is used as constraint.

We observed that nouns are a very common type of POS in the input documents, which explains why using constraints on nouns could not remove false-positive links substantially. For example, in the requirement of "Email client shall open and display a message in a single window", using a constraint on the term "message" will not help remove false-positive links to source code entities containing "receive, send, delete messages". Because ConPOS (using nouns as constraint) will find a matched noun (i.e., "messages") in a class responsible for receiving, sending, and deleting messages. Therefore, using constraints on nouns can provide better accuracy but may not remove significant numbers of false-positive links.

For a deeper analysis, Figures 8 and 9 show the detailed effectiveness results (for all the 100 threshold points) on one of the four subjects Pooka with VSM used as the baseline IR technique for ConPOS. The results show that defining constraints on nouns produced better precision and recall for this subject at some threshold points only in comparison to the baseline. In contrast, for this particular subject, defining constraints on adjectives did not produce better precision and recall; it led to higher precision and recall only at a couple of threshold points.

In all, defining constraints on adjectives or nouns never resulted better results than defining constraints on verbs for the proposed ConPOS approach. Our results reveal that defining constraints could improve the effectiveness of baseline RT approaches for a variety of constraint choices, yet the most effective is to use verbs as constraint as the default option of ConPOS.

5.3. Other Observations

Typically POS taggers tag a term based on the context and grammar of a sentence. For example, a requirement for Pooka is "Email client shall open and display a message in a single window". Here the term "open" is an action and POS tagger tagged it as a verb. `MessageFrame.java` is the source code file responsible to implement the requirement. POS tagger tagged term "open" as an adjective based on the context in the source code file. Thus, in the case of only noun and verbs, removing a term "open" from source code file caused low similarity value between a requirement and source code file. The low similarity value pushed true-positive links down in the ranked list.

A source code document is a mixture of some terms without any context and proper grammar. A POS tagger could incorrectly tag terms because of missing context and proper English sentences. Filtering terms from source code entities based on POS tagger could remove some important semantic information, and thus lead to inaccurate tagging by the POS tagger. Capobianco et al. [10, 11] also observed that in some cases incorrect POS tagging caused poor accuracy.

Therefore, our proposed approach ConPOS only uses POS tagger on requirements and look for the same terms, tagged as verbs, in the source code. We are not concerned whether the same verb is tagged as a verb in a source code entity. For example, in our previous example, ConPOS will recover a link

Table 8: Effectiveness (precision, recall, and MAP) of ConPOS variants using different POS as constraint versus baseline IR-based approaches. CP is short for ConPOS. Boldface indicates a ConPOS approach's result that improves over that of the corresponding IR-based baseline approach.

	Precision (%)				**Recall (%)**				**MAP**			
	iTrust	**Lynx**	**Pooka**	**SIP**	**iTrust**	**Lynx**	**Pooka**	**SIP**	**iTrust**	**Lynx**	**Pooka**	**SIP**
VSM	48.89	65.85	41.31	14.14	23.40	38.26	10.70	13.06	0.27	0.63	0.23	0.12
CP$_{VSM.Vb}$	**67.56**	**74.46**	**63.96**	**29.29**	**27.78**	**41.31**	**17.56**	**19.16**	**0.46**	**0.74**	**0.47**	**0.29**
CP$_{VSM.Adj}$	49.82	65.22	41.85	17.47	26.07	17.33	5.16	11.95	0.27	0.30	0.13	0.11
CP$_{VSM.Nn}$	35.58	62.42	35.98	7.73	29.84	29.50	10.96	18.02	0.27	0.45	0.23	0.13
JSM	34.70	60.30	32.24	18.45	40.58	41.36	12.69	17.07	0.33	0.60	0.17	0.10
CP$_{JSM.Vb}$	**49.92**	**66.68**	**56.67**	**29.43**	**46.25**	**44.76**	**19.94**	**23.75**	**0.48**	**0.69**	**0.41**	**0.27**
CP$_{JSM.Adj}$	33.13	57.30	33.69	20.19	45.52	19.47	6.285	16.11	0.33	0.29	0.10	0.11
CP$_{JSM.Nn}$	29.39	56.48	32.47	20.35	51.30	31.67	12.96	20.83	0.33	0.44	0.18	0.15

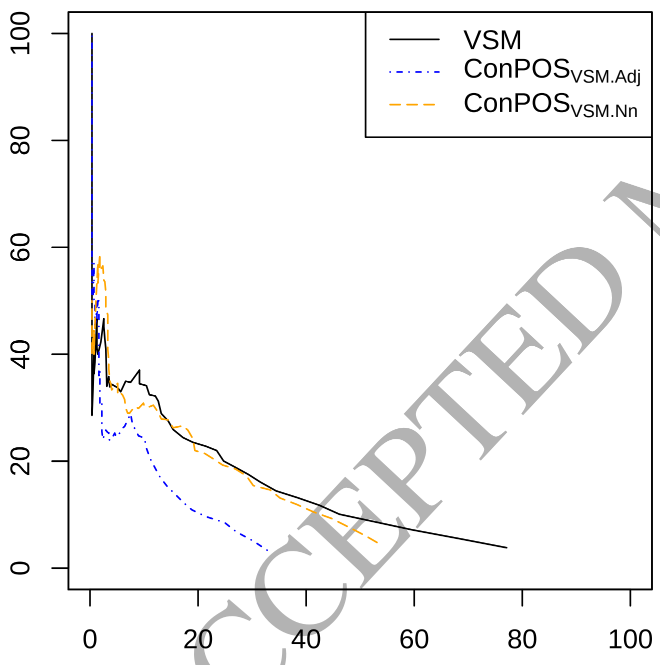


Figure 8: Precision (y axis) and recall (x axis) of ConPOS using VSM and the baseline approach on the Pooka subject, with various types of POS used as constraint for ConPOS and with the threshold t varying from 0.01 to 1 by step of 0.01.

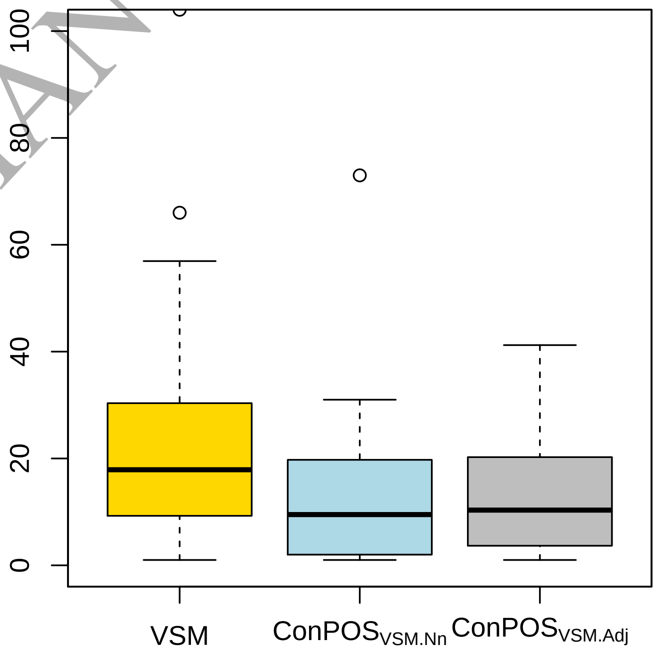


Figure 9: Distribution of average precision (AP) of true-positive links of ConPOS using VSM and the baseline approach on the Pooka subject, with various types of POS used as constraint for ConPOS.

between the requirement and MessageFrame.java because MessageFrame.java contains the term "open". It is quite possible that using such a constraint may not remove all false positive links. Nevertheless, our evaluation study has shown that the constraint-based pruning in ConPOS can drastically improve the effectiveness in recovering trace links over all the baseline RT approaches we considered.

5.4. Comparison with related work

Our conjecture stems from the work of Capobianco et al. [10, 11]. We conjecture that source code entities are mixture of terms without any context and proper grammar. A POS tagger could tag a term incorrectly because of the missing context and grammar. Thus, discarding terms based on incorrect POS tagging could yield semantic information loss. In addition, a verb defines the action of a requirement and removing verbs and only keeping nouns from a requirement could produce more false positive links.

The work presented in this paper is different from the heuristics proposed by Capobianco et al. [10, 11]. We do not discard any POS from the documents because we conjecture all POS contain important semantic information. We use POS of requirements to define some constraints on the trace links to discard false positive trace links. In addition, to the best of our knowledge, this paper is the first attempt to analyze the impact of various POS, i.e., adjectives, nouns, and verbs, on RT.

Table 9 shows the summary of the related work. The column POS Tagging on SD and POS Tagging on TD in Table 9 shows whether the proposed approach use POS tagging on source and target documents. Existing POS taggers are not 100% accurate yet [17]. Thus, as opposed to existing work, we did not tag source code identifiers in this paper. Many researchers have used POS to improve IR techniques accuracy to perform various software maintenance [39, 64, 11, 12, 46, 63, 63]. For example, [65, 66, 61] integrated POS information in a term weighting scheme to improve the accuracy of IR techniques. The column Combination of POS in Table 9 shows whether the existing works have explored the impact of a single, multiple, or a combinations of different POS on recovering trace links. The column Constraint on POS shows whether existing works have used any kind of constraint on a POS. The work presented in this paper is complementary to existing IR-based techniques because it uses current state-of-the-art techniques to recover links and uses a constraint on POS to filter out false-positive links and improve the accuracy.

6. Threats to Validity

Some threats could potentially limit the validity of our experiments. We now discuss potential threats and how we control or mitigate them, following the categorization of validity threats proposed in [52].

Construct validity. Construct validity concerns the relation between theory and observations [52]. To mitigate construct validity threats, in our empirical study, we used widely adopted metrics, precision, recall, and MAP, to assess various baseline techniques and ConPOS as well as the improvement achieved by ConPOS. The ground-truth trace links (traceability matrix) used to evaluate the tracing accuracy is another threat to the construct validity of our results. These links were created by a few students at a local institution, thus the validity of the ground-truth is potentially subject to human biases or mistakes. To mitigate this threat, after the students manually recovered the trace links, two professors verified their results to reduce the possibility of errors in the dataset. The ground-truth links were all approved by the professors before they were used in our studies. Also, all but one of the participants who created the oracles were unaware of the purpose of this process or details about this work. Finally, for iTrust, we used the oracle recovered by the original developers of this software. The developers of iTrust were not aware of the goal of our empirical study either.

Internal Validity. The internal validity of the study is the extent to which a treatment affects change in the dependent variable [54, 52]. The internal validity of our empirical study could be threatened by our choice of verb for the constraint used in ConPOS: constraints using other types of POS could lead to different results. To mitigate this threat, we performed more experiments (i.e., constraints on nouns and adjectives) to examine the impact of the choice of constraints in ConPOS on its effectiveness and found that using verbs as constraint provides the best results. The choice of source code tagger could also impact our empirical study results; hence the use of a different POS tagger might lead to different results. To mitigate this threat, we used POSSE [17], an effective source code POS tagger that outperforms other existing POS taggers, as shown by the authors [17]. However, in general, the POS taggers are not 100% accurate yet. Thus, using more accurate POS taggers in future could lead to better/different results relative to ours.

External Validity. The external validity of the study relates to the extent to which we can generalize its results [54, 52]. Our empirical study is limited to four datasets iTrust, Lynx, Pooka, and SIP. Yet, our approach is applicable to any other software systems. However, we cannot claim that the same results would be achieved with other systems. Different systems with different POS, requirements, reverse engineering tools, and source code may lead to different results. However, the four selected datasets have different POS information, requirements, and source code quality. Our datasets selection reduces the threat to external validity. However, more studies, preferably on industrial datasets, are required to generalize the results of our empirical study.

Another possible threat to external validity lies in the potential biases due to the fact that we used the same set of experimental data (subjects and corresponding ground-truth links) for both the empirical validation of ConPOS (that addressed RQ2 and RQ3) and the study (the exploratory study that addressed RQ1) that motivated the development of ConPOS. However, we only drew observations from the exploratory study to motivate the design of ConPOS, yet ConPOS itself did not directly

Table 9: Comparison between ConPOS and related works. PREQ: POS tagging on requirements document; PSRC tagging on source code; CombP: combinations of POS; PCon: POS used as constraint.

	Adjectives	**Nouns**	**Verbs**	**PREQ**	**PSRC**	**CombP**	**PCon**	**Tasks**
This work	✓	✓	✓	✓		✓	✓	Traceability
[11]		✓		✓	✓			Traceability
[61]	✓	✓	✓					Traceability
[47]		✓		✓	✓			Traceability
[62]		✓	✓	✓	✓			Feature Location
[12]		✓		✓	✓			Feature Location
[46]		✓						Bug Report Assignment
[63]		✓						Tag Recommendation

use any parameters derived from the exploratory study, nor does ConPOS's internal workings depend/rely on any findings from that study.

In addition, we concluded that removing POS can negatively impact the effectiveness of IR-based RT approaches, which is almost contradictory to the conclusion in previous work [11] that discarding some POS information can help improve the accuracy of IR-based RT approaches. A possible threat to our conclusion is that the drastic difference could be partially connected to the fact that we used a different set of subjects and a different POS tagger from what was used in that work. However, we attempt to use the best POS tagger (POSSE) in the literature (as claimed by the authors of the tagger) at the time this work was undertaken (The POSSE authors also acknowledged that this tagger can be inaccurate in tagging source code, yet is still more accurate in doing so than peer taggers at the time). Ideally, we should have used the same experimental dataset as used in [11]. However, we could not use their datasets for requirements to code traceability because the datasets are in Italian.

Conclusion validity. Conclusion validity threat deals with the relation between the treatment and the outcome [52]. We mitigate this threat by paying attention to the distribution of our empirical study results. We verified the data distribution of our results using Shapiro-Wilk test. We further used a non-parametric statistical test (i.e Mann-Whitney), because our performance of the Shapiro-Wilk test revealed that our data is not normally distributed.

7. Conclusion

In this paper, we propose a novel approach, called *ConPOS*, to recover trace links using constraint-based pruning. *ConPOS* uses all major POS categories and applies constraints on top of the recovered trace links for pruning as a filtering process that can significantly improve the effectiveness of IR-based techniques in recovering trace links. To demonstrate that excluding one or more POSs from software artifacts could negatively impact the accuracy of IR-based RT techniques, we conducted an empirical study to verify this conjecture, and our results show that indeed having more semantic information (i.e., all POS) is better for effectively recovering trace links. One of the reasons could be that POS taggers used on source code identifiers are not 100% accurate yet. Thus, using POS on source code to recover trace links may not be very effective.

In addition, we conducted two empirical studies to evaluate the effectiveness of *ConPOS* in recovering trace links compared to existing peer RT approaches. Results show that constraint-based pruning (as the core of ConPOS) can bring significant improvement in every aspect of effectiveness (precision, recall, and MAP of true-positive links) over existing IR-based RT approaches and approaches that removed one or more POSs from input documents. Furthermore, we also found that using verbs as the constraint is the most effective choice of the constraint for ConPOS.

In the future, we plan to integrate ConPOS with other RT approaches, e.g., LSI and LDA, to analyze the improvement made by ConPOS. We also plan to perform a user study to analyze how effectively ConPOS helps developers recover trace links and how much time it saves for developers. To generalize the finding of this paper, we plan to perform more extensive empirical studies, preferably on industrial datasets and with heterogeneous types of software artifacts. We observed in a couple of cases, that only keeping nouns slightly improved recall. In future, we plan to analyze a wide variety of datasets to find pattern characterizing the cases where removing a POS could improve the accuracy of an IR approach.

Acknowledgment

We thank Samir Gupta, Sana Malik, Lori Pollock and K. Vijay-Shanker for providing us their POS tagger tool.

References

[1] O. C. Z. Gotel, C. W. Finkelstein, An analysis of the requirements traceability problem, 1st International Conference on Requirements Engineering (1994) 94–101.

[2] J. Cleland-Huang, M. Heimdahl, J. H. Hayes, R. Lutz, P. Maeder, Trace queries for safety requirements in high assurance systems, in: Requirements Engineering: Foundation for Software Quality, Springer, 2012, pp. 179–193.

[3] J. Hill, S. Tilley, Creating safety requirements traceability for assuring and recertifying legacy safety-critical systems, in: 18th IEEE International Requirements Engineering Conference (RE), IEEE, 2010, pp. 297–302.

[4] O. Gotel, J. Cleland-Huang, J. H. Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, J. Maletic, P. Mäder, Traceability fundamentals, in: J. Cleland-Huang, O. Gotel, A. Zisman (Eds.), Software and Systems Traceability, Springer London, London, 2012, pp. 3–22. doi:10.1007/978-1-4471-2239-5_1.
URL https://doi.org/10.1007/978-1-4471-2239-5_1

[5] T. C. Lethbridge, J. Singer, A. Forward, How software engineers use documentation: The state of the practice, IEEE software 20 (6) (2003) 35–39.

[6] T. Gorschek, M. Svahnberg, Requirements experience in practice: Studies of six companies, in: Engineering and Managing Software Requirements, Springer, 2005, pp. 405–426.

[7] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, E. Merlo, Recovering traceability links between code and documentation, IEEE Transactions on Software Engineering 28 (10) (2002) 970–983. doi:10.1109/TSE.2002.1041053.
URL http://dx.doi.org/10.1109/TSE.2002.1041053

[8] N. Ali, Y.-G. Guéhéneuc, G. Antoniol, Trustrace: Mining software repositories to improve the accuracy of requirement traceability links, IEEE Transactions on Software Engineering 39 (5) (2013) 725–741. doi:10.1109/TSE.2012.71.
URL http://dx.doi.org/10.1109/TSE.2012.71

[9] N. Ali, Y.-G. Gueheneuc, G. Antoniol, Requirements traceability for object oriented systems by partitioning source code, in: Proceedings of the 2011 18th Working Conference on Reverse Engineering, WCRE '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 45–54. doi:10.1109/WCRE.2011.16.
URL http://dx.doi.org/10.1109/WCRE.2011.16

[10] G. Capobianco, A. De Lucia, R. Oliveto, A. Panichella, S. Panichella, On the role of the nouns in ir-based traceability recovery, in: 17th IEEE International Conference on Program Comprehension (ICPC'09), IEEE, 2009, pp. 148–157.

[11] G. Capobianco, A. D. Lucia, R. Oliveto, A. Panichella, S. Panichella, Improving ir-based traceability recovery via noun-based indexing of software artifacts, Journal of Software: Evolution and Process 25 (7) (2013) 743–762.

[12] S. Zamani, S. P. Lee, R. Shokripour, J. Anvik, A noun-based approach to feature location using time-aware term-weighting, Information and Software Technology 56 (8) (2014) 991–1011.

[13] Java language and virtual machine specifications, https://docs.oracle.com/javase/specs/, last accessed: June 2018.

[14] Z. P. Fry, D. Shepherd, E. Hill, L. Pollock, K. Vijay-Shanker, Analysing source code: looking for useful verb–direct object pairs in all the right places, IET software 2 (1) (2008) 27–36.

[15] T. Hoff, C Coding Standard, https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html, last accesses: June 2018 (2008).

[16] N. Ali, Z. Sharafi, Y.-G. Guéhéneuc, G. Antoniol, An empirical study on requirements traceability using eye-tracking, in: 28th IEEE International Conference on Software Maintenance (ICSM), 2012, pp. 191–200.

[17] S. Gupta, S. Malik, L. Pollock, K. Vijay-Shanker, Part-of-speech tagging of program identifiers for improved text-based software engineering tools, in: 21st IEEE International Conference on Program Comprehension (ICPC), 2013, pp. 3–12.

[18] J. Giménez, L. Marquez, SVMTool: A general pos tagger generator based on support vector machines, in: In Proceedings of the 4th International Conference on Language Resources and Evaluation, pp. 43–46.

[19] A. Abadi, M. Nisenson, Y. Simionovici, A traceability technique for specifications, in: The 16th IEEE International Conference on Program Comprehension (ICPC 2008), 2008, pp. 103 –112.

[20] M. Borg, P. Runeson, A. Ardö, Recovering from a decade: A systematic mapping of information retrieval approaches to software traceability, Empirical Softw. Engg. 19 (6) (2014) 1565–1616. doi:10.1007/s10664-013-9255-y.
URL http://dx.doi.org/10.1007/s10664-013-9255-y

[21] A. Marcus, J. I. Maletic, Recovering documentation-to-source-code traceability links using latent semantic indexing, in: Proceedings of 25th International Conference on Software Engineering, IEEE CS Press, Portland Oregon USA, 2003, pp. 125–135.

[22] D. Poshyvanyk, Y.-G. Guéhéneuc, A. Marcus, G. Antoniol, V. Rajlich, Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval, IEEE Transactions on Software Engineering 33 (6) (2007) 420–432. doi:http://doi.ieeecomputersociety.org/10.1109/TSE.2007.1016.

[23] R. Oliveto, M. Gethers, D. Poshyvanyk, A. De Lucia, On the equivalence of information retrieval methods for automated traceability link recovery, in: Proceedings of the 2010 IEEE 18th International Conference on Program Comprehension, ICPC '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 68–71.

[24] M. Borg, P. Runeson, Ir in software traceability: From a bird's eye view, in: 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, 2013, pp. 243–246. doi:10.1109/ESEM.2013.39.

[25] M. Gethers, R. Oliveto, D. Poshyvanyk, A. D. Lucia, On integrating orthogonal information retrieval methods to improve traceability recovery, in: 27th IEEE International Conference on Software Maintenance (ICSM), 2011, pp. 133–142.

[26] S.-H. Cha, Comprehensive survey on distance/similarity measures between probability density functions, International Journal of Mathematical Models and Methods in Applied Sciences 1 (4) (2007) 300–307.

[27] H. U. Asuncion, A. U. Asuncion, R. N. Taylor, Software traceability with topic modeling, in: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1, ACM, 2010, pp. 95–104.

[28] D. Falessi, G. Cantone, G. Canfora, Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques, IEEE Transactions on Software Engineering 39 (1) (2013) 18–44. doi:10.1109/TSE.2011.122.

[29] N. Ali, Y.-G. Guéhéneuc, G. Antoniol, Factors impacting the inputs of traceability recovery approaches, in: A. Zisman, J. Cleland-Huang, O. Gotel (Eds.), Software and Systems Traceability, Springer-Verlag, New York, 2011, Ch. 7.

[30] W. Zhao, L. Zhang, Y. Liu, J. Sun, F. Yang, Sniafl: Towards a static noninteractive approach to feature location, ACM Trans. Softw. Eng. Methodol. 15 (2006) 195–226.

[31] A. D. Lucia, F. Fasano, R. Oliveto, G. Tortora, Recovering traceability links in software artifact management systems using information retrieval methods, ACM Trans. Softw. Eng. Methodol. 16 (4) (2007) 13. doi:10.1145/1276933.1276934.
URL http://doi.acm.org/10.1145/1276933.1276934

[32] G. Antoniol, B. Caprile, A. Potrich, P. Tonella, Design-code traceability recovery: selecting the basic linkage properties, Science of Computer Programming 40 (2-3) (2001) 213–234.

[33] G. Antoniol, B. Caprile, A. Potrich, P. Tonella, Design-code traceability for object-oriented systems, Annals of Software Engineering 9 (1) (2000) 35–58.

[34] C. McMillan, D. Poshyvanyk, M. Revelle, Combining textual and structural analysis of software artifacts for traceability link recovery, in: ICSE Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE'09), IEEE, 2009, pp. 41–48.

[35] M. Grechanik, K. McKinley, D. Perry, Recovering and using use-case-diagram-to-source-code traceability links, in: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, ACM, 2007, pp. 95–104.

[36] D. Diaz, G. Bavota, A. Marcus, R. Oliveto, S. Takahashi, A. De Lucia, Using code ownership to improve ir-based traceability link recovery, in: 21st IEEE International Conference on Program Comprehension (ICPC), 2013, pp. 123–132.

[37] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley, 1999.

[38] B. Dit, L. Guerrouj, D. Poshyvanyk, G. Antoniol, Can better identifier splitting techniques help feature location?, in: 19th IEEE International Conference on Program Comprehension (ICPC), IEEE, 2011, pp. 11–20.

[39] A. Chowdhury, M. C. McCabe, Improving information retrieval systems using part of speech tagging, Tech. rep. (1998).

[40] G. Kowalski, Information retrieval architecture and algorithms, Springer-Verlag New York Inc, 2010.

[41] B. Erol, K. Berkner, S. Joshi, Multimedia thumbnails for documents, in: Proceedings of the 14th annual ACM international conference on Multimedia, MULTIMEDIA '06, ACM, New York, NY, USA, 2006, pp. 231–240.

[42] Y. Sun, P. He, Z. Chen, An improved term weighting scheme for vector space model, in: Proceedings of 2004 International Conference on Machine Learning and Cybernetics, Vol. 3, IEEE, 2004, pp. 1692–1695.

[43] X. Zou, R. Settimi, J. Cleland-Huang, Phrasing in dynamic requirements trace retrieval, in: Computer Software and Applications Conference, 2006. COMPSAC'06. 30th Annual International, Vol. 1, IEEE, 2006, pp. 265–272.

[44] L. H. Etzkorn, L. L. Bowen, C. G. Davis, An approach to program understanding by natural language understanding, Natural Language Engineering 5 (3) (1999) 219–236.

[45] S. L. Abebe, P. Tonella, Natural language parsing of program element names for concept extraction, in: Program Comprehension (ICPC), 2010 IEEE 18th International Conference on, IEEE, 2010, pp. 156–159.

[46] R. Shokripour, J. Anvik, Z. M. Kasirun, S. Zamani, A time-based approach to automatic bug report assignment, J. Syst. Softw. 102 (C) (2015) 109–122. doi:10.1016/j.jss.2014.12.049.
URL http://dx.doi.org/10.1016/j.jss.2014.12.049

[47] G. Capobianco, A. De Lucia, R. Oliveto, A. Panichella, S. Panichella, Traceability recovery using numerical analysis, in: 16th Working Conference on Reverse Engineering (WCRE'09), IEEE, 2009, pp. 195–204.

[48] E. Hill, D. Binkley, D. Lawrie, L. Pollock, K. Vijay-Shanker, An empirical study of identifier splitting techniques, Empirical Software Engineering 19 (6) (2014) 1754–1780.

[49] K. Toutanova, D. Klein, C. D. Manning, Y. Singer, Feature-rich part-of-speech tagging with a cyclic dependency network, in: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1, Association for Computational Linguistics, 2003, pp. 173–180.

[50] M. F. Porter, An algorithm for suffix stripping (1997) 313–316.

[51] D. A. Evans, C. Zhai, Noun-phrase analysis in unrestricted text for information retrieval, in: Proceedings of the 34th annual meeting on Association for Computational Linguistics, Association for Computational Linguistics, Morristown, NJ, USA, 1996, pp. 17–24.

[52] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering: An Introduction, Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[53] N. Juristo, A. M. Moreno, Basics of Software Engineering Experimentation, 1st Edition, Springer Publishing Company, Incorporated, 2010.

[54] B. Kitchenham, S. L. Pfleeger, L. Pickard, P. Jones, D. C. Hoaglin, K. E. Emam, J. Rosenberg, Preliminary guidelines for empirical research in software engineering, IEEE Trans. Software Eng. 28 (8) (2002) 721–734. doi:10.1109/TSE.2002.1027796.

[55] A. Jedlitschka, D. Pfahl, Reporting guidelines for controlled experiments in software engineering, in: 2005 International Symposium on Empirical Software Engineering (ISESE 2005), 17-18 November 2005, Noosa Heads, Australia, 2005, pp. 95–104. doi:10.1109/ISESE.2005.1541818.

[56] N. Ali, W. Wu, G. Antoniol, M. D. Penta, Y.-G. Guéhéneuc, J. H. Hayes, A novel process and its implementation for the multi-objective miniaturization of software, Tech. Rep. EPM-RT-2010-04, Ecole Polytechnique de Montreal, technical Report (2010).

[57] N. Ali, Y. Gueneuc, G. Antoniol, Trustrace: Mining software repositories to improve the accuracy of requirement traceability links, Software Engineering, IEEE Transactions on 39 (5) (2013) 725–741.

[58] J. H. Hayes, G. Antoniol, Y.-G. Guéhéneuc, Prereqir: Recovering pre-requirements via cluster analysis, in: Reverse Engineering, 2008. WCRE '08. 15th Working Conference on, 2008, pp. 165 –174.

[59] E. M. Voorhees, Variations in relevance judgments and the measurement of retrieval effectiveness, Information processing & management 36 (5) (2000) 697–716.

[60] M. D. Smucker, J. Allan, B. Carterette, A comparison of statistical significance tests for information retrieval evaluation, in: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, ACM, 2007, pp. 623–632.

[61] A. Mahmoud, N. Niu, On the role of semantics in automated requirements tracing, Requirements Engineering (2014) 1–20.

[62] W. Zhao, L. Zhang, Y. Liu, J. Sun, F. Yang, Sniafl: Towards a static non-interactive approach to feature location, ACM Transactions on Software Engineering and Methodology (TOSEM) 15 (2) (2006) 195–226.

[63] entagrec.

[64] Z. P. Fry, D. Shepherd, E. Hill, L. Pollock, K. Vijay-Shanker, Analysing source code: looking for useful verb–direct object pairs in all the right places, IET software 2 (1) (2008) 27–36.

[65] C. Lioma, R. Blanco, Part of speech based term weighting for information retrieval, in: Advances in information retrieval, Springer, 2009, pp. 412–423.

[66] A. Mahmoud, N. Niu, Using semantics-enabled information retrieval in requirements tracing: An ongoing experimental investigation, in: 34th IEEE Annual Computer Software and Applications Conference (COMPSAC), IEEE, 2010, pp. 246–247.