**The Impact of Retrieval Direction on IR-Based Traceability Link Recovery**

**Abstract:**

The application of Information Retrieval (IR) techniques to software traceability link recovery has been the focus of many studies. These studies have formulated the task of establishing valid trace links between two types of software-artifacts as a retrieval problem, where one type of artifacts is selected as the set of queries and the other as the corpus. Previous work selected the sets of queries and corpus artifacts for a study up front, therefore pre-imposing a retrieval direction for finding all trace links. This decision was usually made based on intuition or previous work. We argue that the choice of the query and corpus sets (i.e., retrieval direction) can significantly impact the results of IR-based traceability link recovery and should be made with context in mind, as the best choice may be dependent on the properties of each dataset. More than that, we argue thateven within the same system, different traceability links maybe best recovered by using different retrieval directions. In this paper we provide the first evidence to support these claims, showing that retrieval direction can have a significant impact on IR performance for traceability link recovery at both the project and individual link level. Moreover, we propose futurere search directions aimed at predicting the most efficient retrieval direction, as well as approaches leveraging information from both retrieval directions simultaneously.

**Introduction**

SECTION I.

## Introduction

Software traceability is a key consideration for software projects, particularly those that are large or safety-critical. Traceability Link Recovery (TLR) is the software engineering task that seeks to establish links between related software artifacts of *various types* (e.g., source code files, documentation, etc.) within a single software project. Many IR techniques have been proposed to support establishing traceability links [1]. These approaches typically formulate the TLR task as a retrieval problem: given a software artifact S of type A (the *query*), use IR techniques to retrieve artifacts of type B (the *corpus*) that are most similar to it and present them in an ordered list to the developer for inspection. The hope is that artifacts of type B that have a true link to S will be found in the top of that list, such that the developer inspects the least amount of results possible to find all the true links to S.

To employ the IR process for TLR, one must decide which software artifacts will be used as queries and which ones as the corpus. This decision establishes what we refer to as the *retrieval direction*. In some SE tasks supported by IR techniques the retrieval direction is imposed by the nature of the problem. For example, in bug localization, it is imperative that bug reports be the queries and code be the corpus, as the goal is locating bugs in the code and not vice versa. In TLR, however, this is not the case. Trace links are symmetric and the problem does not require pre-imposing a retrieval direction. Given two types of artifacts A and B (e.g., code classes and use cases) for which we want to recover traceability links, one could use either type A artifacts as queries and type B as corpus (retrieval direction *A*→*B*), or type B artifacts as queries and type A as corpus (retrieval direction *B*→*A*) [2].

Previous work has not offered clear guidelines on choosing the retrieval direction for a given dataset in the context of TLR. We argue and prove, however, that choosing one retrieval direction over another can impact the overall TLR results in a system, as also shown by Marcus et al. [2] in a study on linking documentation and code. More than that, when the retrieval direction changes, the properties of the search space change, such as the distribution of the query terms in the corpus documents, the size of the corpus, etc. This could make a trace link easier to retrieve using one direction, while another one may be better served by the opposite retrieval direction. In this paper we leverage this idea and show the first evidence that even within the same system, choosing any one retrieval direction leads to suboptimal results. We also show that the changes in results and their magnitude vary depending on the software system.

Based on our preliminary results, we believe that leveraging retrieval direction could be the next big step in improving IR-based TLR approaches, and therefore overcoming the current performance plateau [3]. To that end, after presenting the results of our study in Section II, we propose and discuss several new research directions that aim at exploiting this information to further the state-of-the-art in Section III.

SECTION II.

## Preliminary Study on the Impact of Retrieval Direction on IR-Based TLR

To determine if the retrieval direction impacts the results produced by IR-based approaches for TLR, we performed a preliminary study where we investigated the effect of changing the retrieval direction in a small dataset. We were interested in answering the following research question:

* **RQ:** Does the choice of retrieval direction influence the results of IR-based traceability link recovery?

### A. Dataset

Our dataset consisted of three software projects - eAnci, eTour, and SMOS - which have been used in previous TLR studies [4], [5]. For each system, the data contains two artifact sets: use cases (UC), which are natural language documents, and code classes (CC), which are Java class files. The number of UC and CC documents, as well as potential and true links for each system are in Table I.

### B. Methodology

We followed the standard preprocessing steps for preparing software artifacts to be used with IR approaches [6]. In particular, for all three systems, and for both CC and UC artifacts, we removed all formatting and special characters, performed stopword removal, split the code identifiers, and stemmed the words. These processed artifacts are then used by the IR engine as either queries or corpus documents. We used Apache Lucene1 as our IR engine, with the standard settings, TF-IDF weights, and cosine similarity.

To study the impact of the retrieval direction, we ran the IR engine for all systems using both the *C C* → *U C* and then the *UC* → *CC* directions. Because we had the golden set of true links for our dataset, we could assess which retrieval direction led to the best results.

To have a more comprehensive answer to our research question, we performed the analysis on two different levels. First, we used the standard IR metrics of precision and recall to understand how the retrieval direction impacts the overall performance of IR-based TLR approaches at the level of a whole system (similar to [2]), but also across systems. We calculated the average precision and recall for all queries in each project at cut points ranging from one to ten. Here, a cut point refers to the number of documents in the result set used in calculating precision and recall (i.e., how many of the top results are retrieved in response to the query). We perform two sets of calculations for each system: in one we use the *U C* → *C C* search direction and in the other we use *CC* → *UC*.

The second type of analysis was done at the level of individual trace links. More specifically, for each true link *tl* in the golden set, between a class *cc* and a use case *uc*, we:

1. Ran the IR engine with the class *cc* as the query and the set of use cases *U C* as the corpus and recorded the position **p1** at which *uc* appears in the ranked list of results (the lower the better).
2. Ran the IR engine with the use case *uc* as the query and the set of code classes *CC* as the corpus and recorded the position **p2** at which *cc* appears in the ranked list of results.
3. Compared **p1** and **p2** and recorded which retrieval direction was best for link *tl*. More specifically, if **pl<p2**, it means that *tl* was easier to retrieve (i.e., was found higher in the results list) using the direction *C C* → *U C*. If, on the other hand, **pl>p2**, *tl* was better served by using *UC* → *CC*. Finally, if **pl=p2**, both directions achieve the same performance in retrieving *tl*.

We then reported the number of true links for which either *C C* → *U C* or *U C* → *C C* performed better, as well as the number of links where they performed the same.

### C. Results

Table II shows the average precision and recall values across all queries for cut points between one and ten, for both retrieval directions and for all three systems. The columns in bold mark the retrieval direction that performed best for a system for precision and recall, respectively. The results suggest that retrieval direction has an impact on both precision and recall values, in all systems. For example, for eAnci, the difference in recall values is 16% on average across all cut points and can reach 23% when considering a cut point of ten. Precision shows even more dramatic differences between search directions. For example, for eAnci, the difference between precisions for the two retrieval directions is 25% on average across all cut points, and up to 33% for the top result (i.e., cut = 1).

The data also suggests that the most efficient retrieval direction is not consistent across projects. For eAnci and SMOS, both precision and recall are higher for one search direction; however, this direction is different for the two systems. Perhaps more interestingly is the case of eTour, which has higher recall for one direction and higher precision in the other. Ultimately there appears to be no universal search direction that is most efficient in terms of precision and recall for all projects; this is an important result but not particularly surprising given the wide range of document properties that projects can have. Our data supports that the claim that retrieval direction impacts IR effectiveness at the project level.

Table III shows our analysis at the level of individual links. Specifically, the total number of valid trace links for which either search direction is the most effective or for which there is no difference in ranking when changing the search direction.

This analysis provides a more surprising result: although both the average precision and recall is much higher for the *CC* → *UC* direction in the eAnci project, there are more individual valid links that are ranked higher when using a use case as a query. Moreover, while *UC* → CC was more effective for SMOS in general, more individual links are ranked higher when using a code class as the query. Also of interest is the fact that while precision and recall had mixed results for eTour when changing the search direction, more than 60% of the valid links are ranked at a higher position when using code classes as queries. These results suggest that, even though each project might have a direction that improves the average precision and recall, efficiency can be further improved on a link-by-link basis. Therefore, this analysis suggests that retrieval direction impacts the efficiency of IR approaches at the individual link level as well.

Furthermore, in order to understand if these results are unique to a particular IR engine, we present the results of the same analysis using Lucene's BM25 approach and similarity in table IV. Overall, results for these two IR implementations differ by only 45 links across all three systems: 2 for eAnci, 18 for eTour, and 25 for SMOS. This suggests that the impact shown in the

SECTION III.

## Future Directions

Motivated by the outcome of our preliminary study, we propose two main research directions that leverage the notion of retrieval direction for IR-based approaches to TLR. The two directions are based on fundamentally different approaches.

### A. Finding the Best Retrieval Direction for a TLR Context

The first research direction is to automatically determine the best retrieval direction to use for TLR in a particular context (i.e., a system or an individual trace link). We envision this as a classification problem, where the two possible retrieval directions represent the classes. Using features in the data, a machine learning algorithm would be trained to automatically recommend the most adequate retrieval direction given the context. One of the research questions to investigate in this direction is what features in the data would be useful for this task. A few basic features to start from could be the number of documents and terms in the two sets of artifacts, the term distribution and usage among the software artifacts, the average size of the documents, etc.

### B. Using Bidirectional Retrieval Information for TLR

The second research direction leverages information about both retrieval directions at the same time - rather than discarding one - to gain more insight into the semantic similarity of the two artifacts in relation to one another and to the rest of the software artifacts. The goal is to discover patterns in this bidirectional information that could indicate whether a link between two artifacts is a true link or not.

We envision constructing machine learning models that use features engineered with retrieval direction in mind, in order to directly classify potential links as either valid or invalid. In particular, we plan on leveraging features based on bidirectional IR rankings using a variety of similarity measures and representations. Additionally, we know from previous research [7] that the quality of a query has a significant impact on the performance of IR-based techniques; therefore, features derived from query quality metrics applied to both documents in a potential link could help identify true links by determining if the artifacts constitute high or low quality queries.

SECTION IV.

## Related Work

For the purpose of this paper, we consider the most relevant work that of Borg et al. who published a systematic mapping study on the use of IR techniques in TLR [1] and therefore puts this study in context. To the best of our knowledge, no previous work has specifically investigated the impact of retrieval direction on the results of IR in TLR at both the project and individual link level. While Marcus et al. [2] applied both retrieval directions using LSI for TLR in a system, the paper does not discuss general guidelines on choosing a retrieval direction for a dataset. More than that, in this paper we analyze the effect of retrieval direction across several projects and two different IR engines, but more importantly at the level of individual links within those projects. We also discuss for the first time potential future research that leverages this effect to further the state-of-the-art.

TLR as an IR retrieval problem was originally introduced by Antoniol et al. [8], who extracted queries from classes in the C++ Library of Efficient Data types and Algorithms (LEDA) project and mapped them to natural language documentation using the Vector Space Model. IR-based TLR has since flourished as a field of research, knowing contributions from many researchers along several directions, such as: investigating the effects of different information retrieval engines such as Latent Semantic Indexing [2], Latent Dirachlet Allocation [9], and BM25 [10]; linking other types of software artifacts, such as UML and documentation [11], use cases and code classes [4], etc.; empirically evaluating tool implementations [12], [13] and human interactions [14]–[15][16]; and augmenting IR with additional sources of information, such as the use of a thesaurus [17], phrasing, and glossaries [18].

Many studies modeled as a retrieval problem follow the process established in [8], using class files as queries to search a corpus constructed from natural language documentation [2]. However, there are also studies leveraging probabilistic retrieval [11], where the retrieval direction is an implementation detail not directly addressed or explained in the paper. Moreover, other approaches side-step directionality by removing it from the process entirely, leveraging term-weighting and co-occurrence [19], or IR-based voting schemes [9] to recover or assess the quality of candidate links. While there is a vast set of related work in this field, we are unaware of any research directly studying the impact of IR retrieval direction to increase retrieval efficiency for TLR.

SECTION V.

## Conclusion

Previous work in the application of IR to TLR has focused on evaluating new IR approaches, alternative sources of information, tools implementing certain approaches, and the empirical evaluation of these tools. However, little to no importance has been given to the impact of the retrieval direction on the performance of IR approaches for TLR.

In this paper we presented preliminary evidence that the retrieval direction can significantly impact IR results for TLR at the level of a system, across systems, and at the level of individual trace links within a system. Moreover, we proposed two novel research directions that leverage this knowledge in order to further improve the performance of IR approaches by choosing the best retrieval direction for a system or link, or using both directions in conjunction to categorize the validity of traceability links.