TraCter: A Tool for Candidate Traceability Link Clustering

Anas Mahmoud
Computer Science and Engineering
Mississippi State University
Mississippi State, MS 39762
amm560@msstate.edu

Nan Niu
Computer Science and Engineering
Mississippi State University
Mississippi State, MS 39762
niu@cse.msstate.edu

Abstract—Automated tracing tools employ information retrieval (IR) methods to recover traceability links between software artifacts. A large body of research is available on the back-end design of such tools, including artifacts indexing and the underlying IR mechanism. In contrast, less attention has been paid to the front-end presentation of the retrieved results. This paper describes TraCter, a result categorization tool with novel search user interfaces. We discuss the key features of TraCter and its potential improvements over previous work.

Keywords—traceability; requirements; information retrieval; result categorization; search user interface

I. INTRODUCTION

Contemporary requirements tracing tools employ information retrieval (IR) methods to automatically generate candidate links. The automated tracing process consists of three steps: indexing, retrieval, and presentation. In the indexing step, input artifacts are converted into more compact forms that are compatible with the underlying IR model [1]. In the retrieval step, IR algorithms, such as latent semantic indexing and vector space model [2], are used to identify a set of candidate links by matching a trace query with artifacts in the software repository. In the presentation step, retrieved candidate traceability links are presented to the human analyst for further validation.

The research literature is primarily concerned with the back-end issues of automated tracing tools, e.g., indexing [1] and retrieving [2]. In contrast, research on presentation of retrieved results is sparse. Lack of effective presentation support for automated tracing tools is an important problem, because the retrieved results must be vetted by a human analyst to determine the final traceability information [3].

A majority of automated tracing tools today adopt a ranked-list presentation, in which all retrieved links are displayed as a sequential list based on the similarity score computed by the IR algorithm. RETRO [2] and ADAMS [4] are among the tools that adopt the ranked-list interface. Although ranked-list is straightforward, the presentation has serious drawbacks. The ranked-list is lengthy, ill-structured, difficult to understand, and more importantly, isolates the retrieved elements from their context in that the relationship between adjacent links cannot be easily inferred [5].

One way to overcome the limitations of ranked-list is providing rich contextual information via search result categorization [6]. Result categorization re-organizes the retrieved elements by grouping topically coherent items together and by presenting descriptive category labels to the human user. The result categories provide associative views for analyst to make connections about the candidate links that would otherwise be invisible. The Poirot tool discussed in [5] clusters retrieved links into result categories.

In this paper, we introduce TraCter (TraCability Link Clustering), a result categorization tool with novel search user interfaces. We next review the research in clustering support for automated tracing. We then discuss TraCter’s features and its potential improvements over previous work.

II. TRACEABILITY LINK CLUSTERING

In their seminal paper, Duan and Cleland-Huang [5] demonstrated the benefits of utilizing clustering in automated tracing. They examined three clustering algorithms: hierarchical agglomerative, k-means, and bisecting divisive, and reported that at reasonable clustering granularity, of 5-6 clusters or higher, no significant difference was observed between the clusters produced by the three algorithms [5]. This is an important finding because there is unlikely to be a clear winner among the many different clustering algorithms when it comes to traceability link categorization.

Fig. 1 shows the cluster-enabled Poirot tool [5], which uses bisecting divisive clustering for its computational efficiency. In order to facilitate human comprehension, Poirot applies the rule of thumb, 7±2 [7], to determine the desired clustering granularity level: an average cluster size therefore should contain 7±2 elements. The generated clusters are presented in multiple pages (cf. Fig. 1). For each cluster, a descriptive label is generated based on its dominant theme, which is defined by terms occurring across a significant number of artifacts within that cluster.

Another key design of Poirot is its capability of overcoming the limitation of monothetic clustering, i.e., clustering based only on a single shared feature [8]. In Poirot, two rounds of clustering are performed and the polythetic clusters [8] are shown in separate tabs (cf. Fig. 1). Poirot has successfully addressed several back-end issues of result categorization tools, most notably the use of bisecting divisive as its underlying clustering algorithm. However, when the presentation side is concerned, we believe there is room for improvement.

III. TRACTER

Fig. 2 shows the TraCter tool that we have developed to address several presentation issues of Poirot, e.g., tab-switching and page-flipping (cf. Fig. 1). Our design intent is to apply novel search user interfaces [6] so as to increase the human analyst’s browsing efficiency. The key features of TraCter are as follows.
Figure 1. Cluster-enabled Poirot [5]

- The 7±2 rule [7] applied to clusters rather than links. Similar to Poirot, TraCter uses the efficient divisive clustering algorithm (O(n) time complexity) and automatically labels clusters. Unlike Poirot returning clusters that have 7±2 links, we confine the number of clusters produced at each clustering iteration to be 7±2. This avoids page-flipping in Poirot. Fig. 2-A (area A) illustrates that each clustering round results in 5 clusters. Note that TraCter produces relatively uniformly sized clusters—a nice property of the underlying divisive clustering algorithm [5].

- Polythetic clusters organized in hierarchies rather than tabs. Poirot, in its current implementation, performs exactly two rounds of polythetic clustering and shows results in separate tabs. TraCter uses a hierarchical tree-view to display polythetic clusters (cf. Fig. 2-A). This not only avoids tab-switching which may cause expensive context-switching, but also allows polythetic clustering to be performed more flexibly than exactly twice.

- Rich contextual cues. TraCter exploits query preview [6], showing how many links are in each category. Query preview, given in parentheses to the right of the category label (cf. Fig. 2-A), provides an information scent of where to go next. In addition to the name of the software artifact/document, the snippet (cf. Fig. 2-B) of a candidate traceability link in TraCter also contains the document header (trimmed in 1 line) and the full path. A mouse-hover over a link pops up the full header information (cf. Fig. 2-C). Such an interface design provides a rich set of contextual cues in one single view.

- Support for collecting traceability information. A novel design in TraCter is the shopping-cart-like checkout area in the upper right corner of the tool (cf. Fig. 2-D). This provides a central place for analysts to view and refine their trace selections. Previous tools like Poirot have not had explicit support toward collecting the final traceability information.

IV. SUMMARY

Automated tracing tools have paid little attention to presenting retrieved links. This paper presents TraCter, a candidate traceability clustering tool with novel search user interfaces. The key features of TraCter are described and its potential improvements over previous work are discussed.

We worked with 12 computer science students to test the usability of TraCter. The preliminary results were encouraging in that the subjects in our study rated “satisfactory” in terms of TraCter’s simplicity, flexibility, browsability, and facilitating traceability in general. In our study, TraCter was compared with a ranked-list tool, but a direct comparison between TraCter and Poirot has yet to be done. Besides conducting more empirical evaluations, our future work also include integrating TraCter with more advanced back-end result categorization engines and with more interactive search user interfaces [6].

REFERENCES