Data & Knowledge Engineering xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Automatic query reformulations for feature location in a model-based family of software products

Francisca Pérez[a,*], Jaime Font[a,b], Lorena Arcega[a,b], Carlos Cetina[a]

[a] SVIT Research Group, Universidad San Jorge, Autovía A-23 Zaragoza-Huesca Km. 299, 50830, Villanueva de Gállego, Zaragoza, Spain
[b] Department of Informatics, University of Oslo, Postboks 1080 Blindern, 0316 Oslo, Norway

ARTICLE INFO

Keywords:
Conceptual modeling
Information retrieval
Feature location
Query reformulation
Software maintenance and evolution
Families of software products

ABSTRACT

No maintenance activity can be completed without Feature Location (FL), which is finding the set of software artifacts that realize a particular functionally. Despite the importance of FL, the vast majority of work has been focused on retrieving code, whereas other software artifacts such as the models have been neglected. Furthermore, locating a piece of information from a query in a large repository is a challenging task as it requires knowledge of the vocabulary used in the software artifacts. This can be alleviated by automatically reformulating the query (adding or removing terms). In this paper, we test four existing query reformulation techniques, which perform the best for FL in code but have never been used for FL in models. Specifically, we test these techniques in two industrial domains: a model-based family of firmwares for induction hobs, and a model-based family of PLC software to control trains. We compare the results provided by our FL approach using the query and the reformulated queries by means of statistical analysis. Our results show that reformulated queries do not improve the performance in models, which could lead towards a new direction in the creation or reconsideration of these techniques to be applied in models.

1. Introduction

Feature location (FL) is known as the process of finding the set of software artifacts that realize a particular functionality of software system. No maintenance activity can be completed without locating in the first place the software artifact (e.g., code) that is relevant to the specific functionality [1]. Since FL is one of the main activities performed during software evolution [1] and up to an 80% of a system's lifetime is spent on the maintenance and evolution of system [2], there is a great demand for FL approaches that can help developers to find relevant software artifacts in a family of software products.

Many of FL approaches use of Information Retrieval (IR) techniques [1,3] such as Latent Semantic Indexing (LSI) [4], Latent Dirichlet Allocation (LDA) [5], and Vector Space Model [6] and involve the formulation of a query in natural language (e.g., by the developer). These techniques are statistical methods used to find a feature's relevant software artifact by analyzing and retrieving words that are similar to a query provided by a user. For example, during FL, a developer formulates a query which describes the feature to be located in the code. The query is then run by the IR technique and a list of ranked software artifacts (e.g., classes or methods) is retrieved.

The performance of the retrieval depends greatly on the textual query and its relationship to the text contained in the software artifacts [7]. Hence, this relationship requires knowledge of the vocabulary of the software artifacts to be searched. This knowledge

* Corresponding author.
E-mail addresses: mfperez@usj.es (F. Pérez), jfont@usj.es (J. Font), larcega@usj.es (L. Arcega), ccetina@usj.es (C. Cetina).

https://doi.org/10.1016/j.datak.2018.06.001
Received 2 April 2017; Received in revised form 6 April 2018; Accepted 6 June 2018

can be difficult to acquire in industrial environments that accumulate a vast amount of software over the years, which often emerges ad hoc using software reuse techniques such as duplication (the "clone-and-own" approach) instead of formalizing the variability among the family of software products. Moreover, in these industrial environments, software maintenance tasks are performed by people who have not participated during the development, so the vocabulary that the people use on the textual query to locate features during maintenance tasks can differ from the vocabulary that was used during the development. Therefore, these differences between the query and the text contained in the software artifacts make the performance of the retrieval worse.

To overcome these differences between the query and the text contained in the software artifacts, other FL approaches [7–9] refine the query using automatic reformulation techniques: expansion or reduction. A short query which obtains not relevant results will likely need an expansion strategy (i.e., adding terms) to improve its performance, whereas a verbose query may need a reduction strategy (i.e., removing terms) since the performance uses to be deteriorated handling long queries [10].

To date the vast majority of work in FL has been focused on improving the performance of the retrieval using automatic query reformulation strategies in code (i.e., by better performance we mean retrieving the relevant software artifacts closer to the top of the list of results). Nevertheless, other software artifacts such as the models have been neglected even though models are the cornerstone in Model-Driven Development approaches to generate code.

To cope with this lack, we evaluate whether automatic query reformulation strategies could improve the results of FL in models. Therefore, the contribution of this paper is twofold.

1. We test four automatic query reformulation techniques (Query reduction, Rocchio query expansion, RSV query expansion and Dice query expansion), which perform best in that field [7] and have never been used to locate features in models. Specifically, we test these techniques in two industrial domains: the model-based product family of the BSH group (www.bsh-group.com) and the model-based product family of CAF (www.caf.net/en).

   The BSH group is one of the largest manufacturers of home appliances in Europe. Its induction division has been producing Induction Hobs (sold under the brands of Bosch and Siemens) for the last 15 years. CAF produces a family of PLC software to control the trains that they have been developing over more than 25 years.
2. We compare the results provided by our FL approach using the query as it is (baseline) with the results provided by the four reformulation techniques.

   The results of this paper suggest that current automatic query reformulation techniques should be reconsidered to be applied in models since we found that using the query as it is leads better results in models than including the query expansion/reduction reformulations. We hope that these results help FL users when they work with models to loss the inertia of applying query reformulation techniques as they would do to locate features in code. Moreover, these results would contribute towards a new direction in the creation of new query reformulation techniques or the modification of the existing ones to improve the location of features in models.

The rest of the paper is structured as follows: Section 2 provides the required background on the automatic query reformulation techniques being compared. Section 3 presents the approach to perform feature location in models. Next, Section 4 presents the evaluation performed, and Section 5 shows the results. Section 6 discusses the results. Section 7 describes the threats to validity. Section 8 reviews the related work. Finally, Section 9 concludes the paper.

2. Query reformulation techniques

Researchers in the field of FL have proposed a large variety of approaches for automatic query reformulations for an initial query. These approaches belong to one of the following two categories [11]: query expansion approaches and query reduction approaches.

Next, we introduce briefly these categories with emphasis on the automatic reformulation strategies that we use for FL in models.

2.1. Query reduction

Longer queries are typically used to express more sophisticated information needs. Nevertheless, longer queries use to include both important information and noise, i.e., terms that serve more to confuse the search engine that support it in its task. Since the performance of most commercial and academic search engines deteriorates while handling longer queries [10], the aim of query reduction is to reduce long queries to shorter.

A conservative automatic query reduction approach that has been previously used in software engineering [7,12] consists of eliminating non-discriminating terms, which are those terms that appear in more than 25% of the documents in the corpus.

2.2. Query expansion

Queries require including terms that fit with the vocabulary of the software artifacts to be searched. Nevertheless, the most critical language issue for performance is that the users often do not use the same words [13].

To deal with this issue, several query expansion techniques have been proposed [13]. Since not all of these techniques can be applied in our work (i.e., model based corpus), we selected three existing techniques using the following criteria: 1) we did not consider techniques that relied on sources of information external to the corpus, like the web, or ontologies; 2) we did not selected techniques that are designed to work for natural language documents as they rely on word relationships that exist in natural language