Science of Computer Programming 76 (2011) 1261–1274

Contents lists available at ScienceDirect

Science of Computer Programming

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

Applying a dynamic threshold to improve cluster detection of LSI☆,☆☆

Pieter van der Spek*, Steven Klusener

Faculty of Sciences, VU University Amsterdam, Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Available online 29 December 2010

Keywords:
Feature extraction
Clustering
Reverse engineering
Software architecture
Latent Semantic Indexing

ABSTRACT

Latent Semantic Indexing (LSI) is a standard approach for extracting and representing the meaning of words in a large set of documents. Recently it has been shown that it is also useful for identifying concerns in source code. The tree cutting strategy plays an important role in obtaining the clusters, which identify the concerns. In this contribution the authors compare two tree cutting strategies: the Dynamic Hybrid cut and the commonly used fixed height threshold. Two case studies have been performed on the source code of Philips Healthcare to compare the results using both approaches. While some of the settings are particular to the Philips-case, the results show that applying a dynamic threshold, implemented by the Dynamic Hybrid cut, is an improvement over the fixed height threshold in the detection of clusters representing relevant concerns. This makes the approach as a whole more usable in practice.

1. Introduction

To developers software often appears as a large number of modules each containing hundreds of lines of code. It is, in general, not obvious which parts of the source code implement a given concern. This fact poses various problems when it comes to maintaining the software. The software records knowledge, expertise, and business rules that may not be available anywhere else than in the source code. Typically, existing documentation is outdated (if it exists at all), the system's original architects are no longer available, or their view is outdated due to changes made by others. So, due to a lack of understanding, maintenance introduces incoherent changes which cause the system's overall structure to degrade [1]. Understanding the system in turn becomes harder any time a change is made to it.

A technique which has been used to provide a view on the system is Latent Semantic Indexing (LSI) [2–4]. This view is used to regain some of the long lost knowledge and aid in the maintenance on the system. LSI is used to identify the associations between words used in source code. The associations are used to cluster the source code documents together. These clusters are interpreted as concerns which reveal the intention of the code [5–9].

The approach as such has been described by various authors [4,6] including the authors of this paper [9], but still a lot of questions remain with respect to choosing various settings. An important aspect of the approach is obtaining the clusters, which provide information on the concerns in the code. The clusters are obtained by having a hierarchical clustering algorithm construct a tree structure called a dendrogram. Subsequently, by cutting the dendrogram the source code documents are grouped into individual clusters.

Usually, the dendrogram is cut using a fixed height threshold. However, there is no such thing as a cutting threshold that works in all circumstances. For instance Kuhn et al. [6] use 0.5, while Maletic et al. [4] use 0.7. The appropriate value,

☆ This work has been carried out as a part of the DARWIN project at Philips Healthcare under the responsibilities of the Embedded Systems Institute. This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

☆☆ Note: Some of the visualizations presented make heavy use of colors. Please obtain a color copy of the article for better understanding.

* Corresponding author.
E-mail addresses: pvdspek@cs.vu.nl (P. van der Spek), steven@cs.vu.nl (S. Klusener).

0167-6423/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2010.12.004

therefore, depends on the particular software system under investigation. The "easiest" way for determining this value is by means of visual inspection of the dendrogram resulting from the hierarchical clustering algorithm, which is difficult when the tree becomes large.

Furthermore, in our experiments at Philips Healthcare it turned out that this threshold results in mostly meaningless clusters as this approach creates one or two disproportionally large clusters and several smaller ones, confirming similar experiences reported in other studies [6,7]. By taking the shape of the dendrogram into account it should be possible to obtain better clusters. This can be done by using a dynamic threshold, which varies in different sections of the tree.

To overcome these deficiencies we have used the Dynamic Hybrid cut [10], which does take the shape of the dendrogram into account. We have performed two case studies on the source code of Philips Healthcare where we applied the approach and compared the outcome of the fixed height threshold to the results obtained by using the Dynamic Hybrid cut.

We make no claims about the generalization of the specific numbers mentioned in this article, but in time they can form part of a broader picture. The method described in this article, however, is more generally applicable and is relevant in other companies and for different software portfolios. In short, the main contributions of this work are as follows.

- We show that the de facto standard for cutting dendrograms, the fixed height cut, is not always the most appropriate choice.
- We evaluate the Dynamic Hybrid cut as an alternative to the de facto standard for cutting dendrograms, the fixed height cut.
- We present the results from two case studies performed at Philips Healthcare comparing the results from the fixed height cut to those of the Dynamic Hybrid cut.

Structure of the paper. In Section 2 we present some related work. Next, in Section 3, we provide an introduction to LSI and its application to source code. Section 4 introduces the Dynamic Hybrid cut, while Section 5 provides the setup for the two case studies. Sections 6 and 7 present the two case studies and their results. Subsequently, in Section 8 we discuss some practical applications of the approach. Finally, in Section 9 we provide some thoughts on possible limitations of the algorithm and our experiments and in Section 10 we present our conclusions.

2. Related work

Many papers report on using LSI for software engineering tasks. Some of the software engineering problems, related to concept location, which have been addressed using LSI are program understanding [4,8,6], high-level concept clones identification [11], conceptual cohesion [12], coupling [13], and traceability (e.g. requirements or documents) [14–16]. Outside the domain of LSI-based approaches, various other approaches have been developed using program slicing [17], dynamic analysis [18], historical information [19], and even simple string pattern matching [20].

A technique which is gaining popularity as an alternative information retrieval technique to LSI is Latent Dirichlet Allocation (LDA) [21–24]. As it does not rely on a hierarchical clustering technique, but rather assigns a probability that a context belongs to a concern, it should be able to identify crosscutting concerns. However, a disadvantage of LDA is that is does not derive any interrelationship between extracted concerns [24,25]. The correlated topic model [26], which builds on LDA, should resolve this issue but has not yet been applied to source code. A hierarchical clustering algorithm does allow for identifying such relationships. Using the Concern Tree they are visualized in an easy-to-understand manner.

Most, if not all, of the approaches using LSI apply clustering and a tree cutting strategy for detecting clusters. An extensive overview of clustering approaches, which are available for software maintenance tasks, is presented in [27]. Clustering approaches have been used in a wide variety of ways to aid software maintenance usually as a means for understanding the (structure of) code. They have been used on their own [28–30] or in combination with other techniques such as formal concept analysis [31,32]. Formal concept analysis on its own turned out to be less suitable for such tasks, especially when applied to large software archives [33]. Furthermore, several researchers have provided methods for evaluating the clusterings made using these algorithms [34,35].

Tree cutting is usually done by means of a single, horizontal cut of the tree structure resulting from the hierarchical clustering. However, often there is no single horizontal cut which accurately reproduces the desired clustering, even when the clusters are easily spotted by means of visual inspection [36]. In this paper, therefore, we have investigated an alternative tree cutting strategy and compared the results to those obtained by using a horizontal cut. Various alternative cutting strategies are available in the literature, such as runt pruning [37], and minimum discrepancy [38]. However, these algorithms use some kind of background knowledge to determine the clusters in the tree. The alternative algorithm evaluated in this paper does not use such knowledge and is therefore ideally suited to be used together with the tree created from the information gathered as part of our approach.

Thus, our work complements existing working using hierarchical clustering as well as our own work presented in [9]. By providing a more flexible tree cutting strategy we overcome the deficiency in existing approaches which use a fixed height threshold even when there is not a single, suitable threshold. The Dynamic Hybrid cut presented in this paper dynamically chooses different thresholds in different parts of the dendrogram resulting in a set of clusters which reflect the structure the dendrogram.