Available online at www.sciencedirect.com
ScienceDirect
Information Processing and Management 43 (2007) 431–444
INFORMATION PROCESSING & MANAGEMENT
www.elsevier.com/locate/infoproman

Supervised categorization of JavaScript™ using program analysis features

Wei Lu [a,*], Min-Yen Kan [b]

[a] Singapore-MIT Alliance, E4-04-10, 4, Engineering Drive 3, Singapore 117576, Singapore
[b] Department of Computer Science, School of Computing, National University of Singapore, Singapore 117543, Singapore

Received 16 May 2006; accepted 25 July 2006

Abstract

Web pages often embed scripts for a variety of purposes, including advertising and dynamic interaction. Understanding embedded scripts and their purpose can often help to interpret or provide crucial information about the web page. We have developed a functionality-based categorization of JavaScript, the most widely used web page scripting language. We then view understanding embedded scripts as a text categorization problem. We show how traditional information retrieval methods can be augmented with the features distilled from the domain knowledge of JavaScript and software analysis to improve classification performance. We perform experiments on the standard WT10G web page corpus, and show that our techniques eliminate over 50% of errors over a standard text classification baseline.

Keywords: Information retrieval; Machine learning; JavaScript; ECMAScript; Program comprehension; Source clone; Program pattern; Software metrics; Program classification; Automated code classification

1. Introduction

As the web has become more interactive, tasks such as form processing, uploading, scripting, applets and plug-ins have transformed the web page into a dynamic application. While a boon to human users, such dynamic aspects of web page scripting and applets impede the machine parsing and understanding of web pages. When such functionality is present, these crucial functionalities are lost if not processed. Pages with JavaScript, Macromedia Flash and other plug-ins are only starting to be indexed and analyzed by web crawlers and indexers. Interactivity in web pages is increasing as more web content is provided using complex content management systems, which use scripting to create a more interactive experience for the human user. Asychronous Javascripting and XML (AJAX) is an example of such an emerging technique. If automated indexers are to keep providing accurate and up-to-date information, methods are needed to glean information about the dynamic aspects of web pages.

* Corresponding author. Tel.: +65 6516 1371.
E-mail addresses: luwei@nus.edu.sg (W. Lu), kanmy@comp.nus.edu.sg (M.-Y. Kan).

0306-4573/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ipm.2006.07.019

To address this problem, we consider a technique to automatically categorize uses of JavaScript, a popular web scripting language. In many web pages, JavaScript realizes many of the dynamic features of web pages. We chose to focus on JavaScript as (1) its code is inlined within an HTML page and (2) embedded JavaScript often interacts with other static web page components (unlike applets and plug-ins). We leverage on both of these key properties in our analysis.

An automatic categorization of JavaScript can assist both indexing software to accurately model web pages' functionality and requirements and browsers to selecting allow certain scripting functions to run and to disable others. Pop-up window blocking, which has been extensively researched, is just one of the myriad uses of JavaScript that would be useful to categorize. Such software can assist automated web indexers to report useful information to search engines and allow browsers to block annoying script-driven features of web pages from end users.

We start with a system that employs traditional text categorization metrics as a baseline. Although the resulting baseline system performs reasonably, we pursue a machine learning framework that draws on features from text categorization, program comprehension and code metrics to improve performance. We show that the incorporation of features that leverage knowledge of the JavaScript language together with program analysis improves categorization accuracy. We conduct evaluation of our methods on the widely used WT10G corpus (Hawking, 2004) to validate our claims and show that the performance of our system eliminates over 50% of errors over the baseline.

In the next section, we examine earlier attempts at source code categorization and discuss how features of the JavaScript language and techniques in program analysis can assist in categorization. We then present our methods that distills features for categorization from the principles of program analysis. We describe our experiment and analysis, and conclude by discussing our manual analysis and future directions of our work.

2. Background

A survey of previous work shows that the problem of automated computer software categorization is relatively new. We believe that this is due to two reasons. First, programming languages are generally designed to be open-ended and largely task-agnostic. Languages such as FORTRAN, Java and C are suitable for a very wide range of tasks and attempting to define a fixed categorization scheme for programs is largely subjective, and likely to be ill-defined. Second, the research fields of textual information retrieval (IR) and program analysis have largely developed independently of each other. We feel that these two fields have a natural overlap which can be exploited.

Unlike natural language texts, program source code is unambiguous to the compiler and has exact syntactic structures. This means that syntax plays an important role that needs to be captured. This has been largely ignored by text categorization research.

Ugurel et al. (2002) and Krovetz et al. (2003) are perhaps the first work that uses IR methods to attack this problem. They employ support vector machines for source code classification in a two-phase process consisting of programming language classification followed by topic classification. In the second, topic classification task, they largely relied on each software projects' README file and comments. From the source code itself, only included header file names were used as features. We believe a more advanced treatment of the source code itself can assist such topic classification. These features, including syntactic information and some language-specific semantic information, could be important and useful for classification. Other recent work in categorizing web pages (Wong & Fu, 2000) has revived interest in the structural aspect of text. One hypothesis of this work is that informing these structural features with knowledge about the syntax of the programming language can improve source code classification.

Program analysis, in which formal methods are employed, has developed into many subfields. The subfield of program comprehension develops models to explain how human software developers learn to comprehend existing code (Mathias, James, Cross, Hendrix, & Barowski, 1999). These models show that developers use both top–down and bottom–up models in their learning process (von Mayrhauser & Vans, 1994). Top–down models imply that developers may use a model of program execution to understand a program. Formal analysis via simulated code execution (Blazy & Facon, 1998) may yield evidence for automated categorization techniques.