ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Implementation on Quality Analysis in Web Applications to Develop Specification and Duplication Mining

M.Mani Mekalai
Sri Krishna arts and Science College, India
manimekalai.m.v.a@gmail.com

Abstract

We propose an approach to Analysis of modern web applications and detect duplicated blocks of code to quality improvement and specification in Web sites and on the analysis of both the page structure, implemented by specific sequences of HTML tags, and the displayed content. In addition, for each pair of dynamic pages we also consider the similarity degree of their scripting code. The similarity degree of two pages is computed using different similarity metrics for the different parts of a web page based on the code duplication string edit distance. We have implemented a prototype to automate the duplicate detection process on web applications developed using technology and used it to validate our approach in a case study. In this system an approach to duplicate analysis for Web applications has been proposed together with a prototype implementation for web pages. Our approach analyzes the page structure, implemented by specific sequences of HTML tags, and the content displayed for both dynamic and static pages. Indeed, we plan to exploit the results of the duplicate analysis method to support web application reengineering activities

Keywords: Code refactoring, prototype implementation, reengineering, trustworthiness and cocitation degree

Introduction

Code refactoring are similar program structures of considerable size and significant similarity. Several studies suggest that as much as 20-50 percent of large software systems consist of cloned code. Knowing the location of clones Helps in program understanding and maintenance. Some clones can be removed with refactoring, by replacing them with function calls or macros, or we can use unconventional metalevel techniques such as Aspect-Oriented Programming or XVCL to avoid the harmful effects of clones.Refactoring is an active area of research, with a multitude of refactoring detection techniques been proposed in the literature. One limitation of the current research on code clones is that it is mostly focused on the fragments of duplicated code (we call them simple clones), and not looking at the big picture where these fragments of duplicated code are possibly part of a bigger replicated program structure. We call these larger granularity similarities structural clones. Locating structural clones can help us see the forest from the trees, and have significant value for program understanding, evolution, reuse, and reengineering. Refactoring tools produce an overwhelming volume of simple refactoring' data that is difficult to analyze in order to find useful clones. This problem prompted different solutions that are related to our idea of detecting structural clones. Some clone detection approaches target large-granularity clones such as similar files, without specifying the details of the low-level similarities contained inside them.

Existing System Drawbacks

Our previous work suggested that programmers make mistakes in error-handling code, perhaps because programmers do not reason properly about uncommon code paths (such as those through catch blocks).We surmise that a candidate that is adhered to on common paths but violated on uncommon paths is thus more likely a true specification, as the violations are more likely to be bugs.That statically predicts the likelihood that a path will be executed when its enclosing method is called (its predicted frequency)We observe that the hypothesized pattern holds for the adhering and violating traces of the candidatesOther research presented human-defined code readability metric; more readable code is correlated with fewer errors.Reapplying

ISSN: 2277-9655
Impact Factor: 1.852

the logic from above, we hypothesize that the true specification's adhering traces are more readable than its violating traces (containing an error), and that such a distinction might not hold for the false candidate

Proposed System

This project proposes an approach for detecting duplicates in web sites and web applications, obtained tailoring the existing methods to detect duplicates in traditional software systems. The approach has been assessed performing analysis on several web sites and web applications.Maintaining software systems are getting more complex and difficult task, as the scale becomes larger. It is generally said that code duplicate is one of the factors that make software Maintenance difficult. This project also develops a maintenance support environment, which visualizes the code duplicate information and also overcomes the limitation of existing tools.

Proposed Methodology

Extend Co-Citation Algorithm:

A Cocitation algorithm that extends the traditional Cocitation concepts. The Cocitation analysis has been used to measure the similarity of papers, journals, or authors for clustering. For a pair of documents p and q, if they are both cited by a common document, documents p and q is said to be co cited Then number of documents that cite both p and q is referred to as the cocitation degree of documents p and q. The similarity between two documents is measured by their cocitation degree. This type of analysis has been shown to be effective in a broad range of disciplines, ranging from author cocitation analysis of scientific sub fields to journal cocitation analysis. In the context of the web, the hyperlinks are regarded as citations between the pages. If a web page p has a hyperlink to another page q, page q is said to be cited by page p. In this sense, citation and cocitation analyses are smoothly extended to the web page hyperlink analysis.

The extended cocitation algorithm is presented with a new page source. It is constructed as a directed graph with edges indicating hyperlinks and nodes representing the following pages.

- page u
- Up to B parent pages of u and up to BF child pages of each parent page those are different from u
- Up to F child pages of u and up to FB parent pages of each child page those are different from u

The parameters B, BF, and FB are used to keep the page source to a reasonable size. Before giving the Extended Cocitation algorithm for finding relevant pages, the following concepts are defined

Truth Finder Algorithm

We can infer the website trustworthiness if we know the fact confidence and vice versa. As in Authority-Hub analysis and Page Rank, TRUTHFINDER adopts an iterative method to compute the trustworthiness of websites and confidence of facts. Initially, it has very little information about the websites and the facts. At each iteration, TRUTHFINDER tries to improve its knowledge about their trustworthiness and confidence, and it stops when the computation reaches a stable state.

Truth Finder Algorithm

Algorithm 1: TRUTHFINDER

Input: The set of web sites W, the set of facts F, and links between them.

Output: Web site trustworthiness and fact confidence.

Calculate matrices **A** and **B**

for each $w \in W$ /* *setting initial state* */

$t(w) \leftarrow t_0$

$\tau(w) \leftarrow -\ln(1 - t(w))$

repeat /* *iterative computation* */

$\vec{\sigma}^* \leftarrow \mathbf{B}\vec{\tau}$

compute $\vec{s}$ from $\vec{\sigma}^*$

$\vec{t}' \leftarrow \vec{t}$ /* *make a copy of* $\vec{t}$ */

$\vec{t} \leftarrow \mathbf{A}\vec{s}$

compute $\vec{\tau}$ from $\vec{t}$

until cosine similarity of $\vec{t}$ and $\vec{t}'$ is greater than $1 - \delta$

As in other iterative approaches TRUTHFINDER needs an initial state. We choose the initial state in which all websites have uniform trustworthiness t0. (t0 should be set to the estimated average trustworthiness, such as 0.9.) From the website trustworthiness TRUTHFINDER can infer the confidence of

facts, which are very meaningful because the facts supported by many websites are more likely to be correct. On the other hand, if we start from a uniform fact confidence, we cannot infer meaningful trustworthiness for websites. Before the iterative computation, we also need to calculate the two matrices A and B, as defined.

They are calculated once and used at every iteration. In each step of the iterative procedure, TRUTHFINDER first uses the website trustworthiness to compute the fact confidence and then recomputed the website trustworthiness from the fact confidence. Each step only requires two matrix operations and Conversions between tðwÞ and _ðwÞ and between sðfÞ and __ðfÞ. The matrices are stored in sparse formats, and the computational cost of multiplying such a matrix and a vector is linear with the number of nonzero entries in the matrix. TRUTHFINDER stops iterating when it reaches a stable state. The stableness is measured by how much the trustworthiness of websites changes between iterations. If t(w) ! Only changes a little after an iteration (measured by cosine similarity between the old and the new t(w), then TRUTHFINDER will stop.

Modules Description

1. **Web URL Identification**

In computing, a Uniform Resource Locator (URL) is a type of **Uniform Resource Identifier** (URI) that specifies where an identified resource is available and the mechanism for retrieving it. In popular usage and in many technical documents and verbal discussions it is often, imprecisely and confusingly, used as a synonym for uniform resource identifier. The confusion in usage stems from historically different interpretations of the semantics of the terms involved. In popular language a URL is also referred to as a Web address.

2. Information Extraction and Parsing

The HTML Parsing module is a class for accessing HTML as tokens. An HTML Parsing object gives you one token at a time, much as a file handle gives you one line at a time from a file. The HTML can be tokenized from a file or string. The tokenizer decodes entities in attributes, but not entities in text. A program that extracts information by working with a stream of tokens doesn't have to worry about the peculiarity of entity encoding, whitespace, quotes, and trying to work out where a tag ends.

Regular expressions are powerful, but they're a painfully low-level way of dealing with HTML. The system processes the spaces and new lines, single and doubles quotes, HTML comments, and a lot more. The next step up from a regular expression is an HTML tokenize.

In this chapter, we'll use HTML Parser to extract information from HTML files. Using these techniques, you can extract information from any HTML file, and never again have to worry about character-level trivia of HTML markup. And automatic passage extraction methods from the body may be worthwhile. Implications of the findings for aids to summarization, and specifically the Text

3. Template Finding & Training

TRUTHFINDER also finds some large trustworthy bookstores such as A1 Books, which provides 86 of 100 books with an accuracy of 0.878. Please notice that TRUTHFINDER uses no training data, and the testing data is manually created by reading the authors' names from book covers. Therefore, we believe TRUTHFINDER performs iterative computation to find out the set of authors for each book. In order to test its accuracy, we randomly select 100 books and manually find out their authors. We find the image of each book and use the authors on the book cover as the standard fact

4. Clone Comparison and Mining

Finally, we perform an interesting experiment on finding trustworthy websites. It is well known that Google (or other search engines) is good at finding authoritative websites. However, do these websites provide accurate information? To answer this question, we compare the online bookstores that are given highest ranks by Google with the bookstores with highest trustworthiness found by Levenshtein distance uses iterative methods to compute the website trustworthiness and fact confidence, which is widely used in many link analysis approaches, . The common feature of these approaches is that they start from some initial state that is either random or uninformative. Then, at each iteration, the approach will improve the current state by propagating information (weights, probability, trustworthiness, etc.) through the links.

Presentation and Discussion of Finding

CODE clones are similar program structures of considerable size and significant similarity. Several studies suggest that as much as 20-50 percent of large software systems consist of cloned code. Knowing the location of clones helps in program understanding and maintenance. Some clones can be removed with refactoring, by replacing them with function calls or macros, or

ISSN: 2277-9655
Impact Factor: 1.852

we can use unconventional metalevel techniques such as Aspect-Oriented Programming or XVCL to avoid the harmful effects of clones.

Cloning is an active area of research, with a multitude of clone detection techniques been proposed in the literature. One limitation of the current research on code clones is that it is mostly focused on the fragments of duplicated code (we call them simple clones), and not looking at the big picture where these fragments of duplicated code are possibly part of a bigger replicated program structure.We call these larger granularity similarities structural clones. Locating structural clones can help us see the forest from the trees, and have significant value for program understanding, evolution, reuse, and reengineering.

Clone detection

The limitation of considering only simple clones is known in the field. The main problem is the huge number of simple clones typically reported by clone detection tools. There have been a number of attempts to move beyond the raw data of simple clones. It has been proposed to apply classification, filtering, visualization, and navigation to help the user make sense of the cloning information. Another way is to detect clones of larger granularity than code fragments. For example, some clone detectors can detect cloned files, while others target detecting purely conceptual similarities using information retrieval methods rather than detecting simple clones

Testing and clustering

Document clustering (also referred to as Text clustering) is closely related to the concept of **data clustering**. Document clustering is a more specific technique for unsupervised document organization, automatic **topic** extraction and fast **information retrieval** or filtering.

A **web search engine** often returns thousands of pages in response to a broad query, making it difficult for users to browse or to identify relevant information. Clustering methods can be used to automatically group the retrieved documents into a list of meaningful categories, as is achieved by Enterprise Search engines such as **Northern Light** and **Vivisimo** or open source software such as **Carrot2**.

Evaluation tree merging

According to our page generation model, data instances of the same type have the same path from the root in the DOM trees of the input pages. Thus, our algorithm does not need to merge similar subtrees from different levels and the task to merge multiple trees can be broken down from a tree level to a string level. Starting from root nodes <html> of all input DOM trees, which belong to some type constructor we want to discover, our algorithm applies a new multiple string alignment algorithm to their first-level child nodes. There are at least two advantages in this design.

First, as the number of child nodes under a parent node is much smaller than the number of nodes in the whole DOM tree or the number of HTML tags in a Webpage, thus, the effort for multiple string alignment here is less than that of two complete page alignments in RoadRunner .

Second, nodes with the same tag name (but with different functions) can be better differentiated by the subtrees they represent, which is an important feature not used in EXALG. Instead, our algorithm will recognize such nodes as peer nodes and denote the same symbol for those child nodes to facilitate the following string alignment.

Third, The string alignment step, we conduct pattern mining on the aligned string S to discover all possible repeats (set type data) from length 1 to length jSj=2. After removing extra occurrences of the discovered pattern (as that in DeLa), we can then decide whether data are an option or not based on their occurrence vector, an idea similar to that in EXALG .

The four steps, peer node recognition, string alignment, pattern mining, and optional node detection, involve typical ideas that are used in current research on Web data extraction. However, they are redesigned or applied in a different sequence and scenario to solve key issues in page-level data extraction.

Testing and Evaluation of page level

Page classification has been addressed with different objectives and methods. Most work concerned form classification methods that are aimed at selecting an appropriate reading method for each form to be processed. Other approaches address the problem of grouping together similar documents in business environments, for instance separating business letters from technical papers In the last few years the classification of pages in journals and books received more attention. An important aspect of page classification is the features that are extracted from the page and used as input to the classifier.

Sub-symbolic features, like the density of black pixels in a region, are computed directly from the image. Symbolic features, for instance the number of horizontal lines, are extracted from a segmentation of the image. Structural features

ISSN: 2277-9655
Impact Factor: 1.852

(e.g. relationships between objects in the page) can be computed from a hierarchical description of the document. Textual features, for instance presence of some keywords, are obtained from the text in the image recognized by an Optical Character Recognition program

Testing and Evaluation of record level

The automatically generated wrapper described in Zhao et al. for instance extracts search result records (SRRs) based on the HTML tag structures. They use learned tag paths pointing to the first (tag) line of candidate records. Their main assumption is that SRRs are usually located in the same sub-tree and its tag path follows certain patterns. Next, they identify the data records in a sub-tree based on learned separators tags and finally generate a regular expression consisting of a path and multiple separators.

The regular expression is evaluated on the tag level which requires that the page must be parsed. To compensate path and separator variations, a wrapper is learned from multiple result pages. Compared to our approach the wrapper extracts the data records as a whole (record level) and does not focus on the attributes contained in the data records (attribute level). Additionally, their approach requires a parser, which also corrects the source to apply the path expressions learned from the parse tree.

Path variations can only be handled by learning a wrapper from multiple result pages, i.e. five result pages and one non-result page are needed in the wrapper generation phase. In contrast, ViPER needs only one result page to generate a wrapper. In Crescenzi et al , union-free regular expressions are deduced, which cannot capture the full diversity of structures presented in HTML. Wang and Lochovsky [2003] propose a system which applies a deduced regular expression for extracting data records from documents. Here labeling can only be carried out after all records have been expensively aligned, i.e. streaming based processing is impossible. All systems mentioned do not support streaming based web content extraction

Comparative study with graph

Our experiment assumes the following situation: A developer starts building an application and uses classes from a library l that are unknown to her. To help the developer avoid bugs due to incorrect usage of those classes, her IDE supports lightweight type state verification. Whenever the developer changes a method that uses classes of l for which a specification is available, the IDE launches the type state verifier. The verifier then analyzes all changed methods and looks for incorrect usage of classes; if it finds a violation, it is presented to the user. Obviously, we would like to catch as many defects and report as few as possible.

Among the first approaches that specifically mine models for classes is the work by Whaley et al. Their technique mines models with anonymous states and slices models by grouping methods that access the same fields. Mine so-called extended finite state machines with anonymous states. To compress models, the gk-tail algorithm merges states that have the same k-future. In terms of static techniques, there is also a huge number of different approaches.

SVM	LEVENSHTEIN	PROPOSED
75.2	60.73	96.26
72.3	58.2	95.94

Conclusions

In this system an approach to clone mining for Web applications has been proposed together with a prototype implementation for dynamic web pages. Our approach analyzes the page structure, implemented by specific sequences of HTML tags, and the content displayed for both dynamic and static pages. Moreover, for a pair of dynamic web pages we also consider the similarity degree of their source. The similarity degree can be adapted and tuned in a simple way for different web applications in one- to-many. We have reported the results of applying our approach and tool in a case study. The results have confirmed that the lack of analysis and design of the Web application has effect on the duplication of the page and code clones are mined in the proposed approach.

In particular, these results allowed us to identify some common features for the multiple code clones and block and their multiple occurrence and influence that could be integrated, by mining the duplications. Moreover, the clone mining of the dynamic web pages , in one-to-many fashion enabled to acquire information to improve the general quality and conceptual/design of the database of the web application. Indeed, we plan to exploit the results of the clone mining method to support web application reengineering activities.

Future Enhancement

Future research on Web data extraction focuses on comparing the contents appearing on

the page as well as the code to measure the standard and originality of the web page. However, they are redesigned or applied in a different sequence and scenario to solve key issues in page-level data extraction and comparison to the code of web site and its contents to find the fake and the real.Implementing good visualizations for higher-level similarities is currently underway . Analysis of clones can also be much facilitated by querying the database of clones. We have already developed a mechanism of creating a relational database of structural clones' data and a query system to facilitate the user in filtering the desired information. Currently, our detection and analysis of similarity patterns is based only on the physical location of clones. With more knowledge of the semantic associations between clones, we can better perform the system design recovery. Using tracing techniques to find associations between classes and methods, we can automate and build a clearer picture of the similarity in process flows within a system to further aid to the user in design recovery.

Acknowledgment

I hereby extend profound and sincere thanks to Mrs.S.Rajanandini M.Sc,P.Phill., Guide for this thesis, Department of Computer Application, Sri Krishna Arts and Science College, Coimbatore for his valuable guidance, constant support, patience and inspiration for completion of the thesis. I am thankful to my respectable parents Mr.R.Murugaraj and Mrs.M.Vijiyalakshmi, my great husband Mr.A.AnandhaKumar and my two kids, as without their support; it would not have been possible to complete the thesis.

References

[1] *Arasu and H. Garcia-Molina, "Extracting Structured Data from Web Pages," Proc. ACM SIGMOD, pp. 337-348, 2003.*

[2] *C.-H. Chang and S.-C. Lui, "IEPAD: Information Extraction Based on Pattern Discovery," Proc. Int'l Conf. World Wide Web (WWW-10), pp. 223-231, 2001.*

[3] *C.-H. Chang, M. Kayed, M.R. Girgis, and K.A. Shaalan, "Survey of Web Information Extraction Systems," IEEE Trans. Knowledge and Data Eng., vol. 18, no. 10, pp. 1411-1428, Oct. 2006.*

[4] *V. Crescenzi, G. Mecca, and P. Merialdo, "Knowledge and Data Engineerings," Proc. Int'l Conf. Very Large Databases (VLDB), pp. 109-118, 2001.*

[5] *C.-N. Hsu and M. Dung, "Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web," J. Information Systems, vol. 23, no. 8, pp. 521-538, 1998.*

[6] *N. Kushmerick, D. Weld, and R. Doorenbos, "Wrapper Induction for Information Extraction," Proc. 15th Int'l Joint Conf. Artificial Intelligence (IJCAI), pp. 729-735, 1997.*

[7] *A.H.F. Laender, B.A. Ribeiro-Neto, A.S. Silva, and J.S. Teixeira, "A Brief Survey of Web Data Extraction Tools," SIGMOD Record, vol. 31, no. 2, pp. 84-93, 2002.*

[8] *B. Lib, R. Grossman, and Y. Zhai, "Mining Data Records in Web pages," Proc. Int'l Conf. Knowledge Discovery and Data Mining (KDD), pp. 601-606, 2003.*

[9] *I. Muslea, S. Minton, and C. Knoblock, "A Hierarchical Approach to Wrapper Induction," Proc. Third Int'l Conf. Autonomous Agents (AA '99), 1999.*

[10] *K. Simon and G. Lausen, "ViPER: Augmenting Automatic Information Extraction with Visual Perceptions," Proc. Int'l Conf. Information and Knowledge Management (CIKM), 2005.*

[11] *J. Wang and F.H. Lochovsky, "Data Extraction and Label Assignment for Web Databases," Proc. Int'l Conf. World Wide Web (WWW-12), pp. 187-196, 2003.*

[12] *Y. Yamada, N. Craswell, T. Nakatoh, and S. Hirokawa, "Testbed for Information Extraction from Deep Web," Proc. Int'l Conf. World Wide Web (WWW-13), pp. 346-347, 2004.*

[13] *W. Yang, "Identifying Syntactic Differences between Two Programs," Software—Practice and Experience, vol. 21, no. 7, pp. 739- 755, 1991.*

[14] *Y. Zhai and B. Liu, "Web Data Extraction Based on Partial Tree Alignment," Proc. Int'l Conf. World Wide Web (WWW-14), pp. 76-85, 2005.*

[15] *H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu, "Fully Automatic Wrapper Generation for Search Engines," Proc.*

Int'l Conf. World Wide Web (WWW), 2005.

[16]Aversano, L., Canfora, G., De Lucia, A., and Gallucci, P., 2001. Web Site Reuse: Cloning and Adapting. Proc. Of 3rd International Workshop on Web Site Evolution, Florence, Italy, IEEE CS Press, pp. 107-111.

[17]De Lucia, A., Scanniello, G., and Tortora, G., 2004."Identifying Clones in Dynamic Web Sites Using Similarity Thresholds," Proc. Intl. Conf. on Enterprise Information Systems (ICEIS'04), pp.391-396.

[18]Di Lucca, G. A., Di Penta, M., Fasilio, A. R., and Granato, P., 2001. "Clone analysis in the web era: An approach to identify cloned web pages," Seventh IEEE Workshop on Empirical Studies of Software Maintenance (WESS), pp. 107–113.

[19]Di Lucca, G. A., Di Penta, M., and Fasolino, A. R., 2002. An Approach to Identify Duplicated Web Pages. Proc. of 26th Annual International Computer Software and Application Conference (COMPSAC'02), Oxford, UK, IEEE CS Press, pp. 481-486.

[20]Kamiya, T., Kusumoto, S., and Inoue, K., 2002. CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code. IEEE Transactions on Software Engineering, 28(7), pp. 654-670.

[21]Kapser, C., and Godfrey, M. W., 2003 "Toward a taxonomy of clones in source code: A case study," In Evolution of Large Scale Industrial Software Architectures, 2003.

[22]Lanubile, F. and Mallardo, T., 2003. Finding Function Clones in Web Application. In Proc. of 7thEuropean Conference on Software Maintenance and Reengineering, Benevento, Italy, IEEE CS Press,pp. 379-386.

[23]Marcus, A., and Maletic, J. I., 2001, "Identification of High-Level Concept Clones in Source Code," Proc. Automated Software Engineering, pp. 107-114.

[24]Ricca, F. and Tonella, P., 2003. Using Clustering to Support the Migration from Static to Dynamic Web Pages. Proc. of 11th International Workshop on Program Comprehension, Portland, Oregon, IEEE CS Press, pp. 207-216.