International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 01 Issue 05
June 2014

Diversify Are Queries Over Xml Data Words

[1]V.Shiva Kumar, Asst. Professor
[2]Dr.Daya Gupta, Professor

ABSTRACT:

While keyword query empowers ordinary users to search vast amount of data, the ambiguity of keyword query makes it difficult to effectively answer keyword queries, especially for short and vague keyword queries. To address this challenging problem, in this paper we propose an approach that automatically diversifies XML keyword search based on its different contexts in the XML data. Given a short and vague keyword query and XML data to be searched, we first derive keyword search candidates of the query by a simple feature selection model. And then, we design an effective XML keyword search diversification model to measure the quality of each candidate. After that, two efficient algorithms are proposed to incrementally compute top-k qualified query candidates as the diversified search intentions. Two selection criteria are targeted: the k selected query candidates are most relevant to the given query while they have to cover maximal number of distinct results. At last, a comprehensive evaluation on real and synthetic data sets demonstrates the

effectiveness of our proposed diversification model and the efficiency of our algorithms.

EXISTING SYSTEM:

- The problem of diversifying keyword search is firstly studied in IR community. Most of them perform diversification as a post-processing or reranking step of document retrieval based on the analysis of result set and/or the query logs. In IR, keyword search diversification is designed at the topic or document level.
- Liu et al. is the first work to measure the difference of XML keyword search results by comparing their feature sets. However, the selection of feature set is limited to metadata in XML and it is also a method of post-process search result analysis.

DISADVANTAGES OF EXISTING SYSTEM:

- When the given keyword query only contains a small number of vague keywords, it would become a very challenging problem to derive the user's search intention due to the high ambiguity of this type of keyword queries.
- Although sometimes user involvement is helpful to identify search intentions of keyword queries, a user's interactive process may be time-consuming when the size of relevant result set is large.
- It is not always easy to get these useful taxonomy and query logs. In addition, the diversified results in IR are often modeled at document levels.
- A large number of structured XML queries may be generated and evaluated.
- There is no guarantee that the structured queries to be evaluated can find matched results due to the structural constraints;
- The process of constructing structured queries has to rely on the metadata information in XML data.

PROPOSED SYSTEM:

- To address the existing issues, we will develop a method of providing diverse keyword query suggestions to users based on the context of the given keywords in the data to be searched. By doing this, users may choose their preferred queries or modify their original queries based on the returned diverse query suggestions.
- To address the existing limitations and challenges, we initiate a formal study of the diversification problem in XML keyword search, which can directly compute the diversified results without retrieving all the relevant candidates.
- Towards this goal, given a keyword query, we first derive the co-related feature terms for each query keyword from XML data based on mutual information in the probability theory, which has been used as a criterion for feature selection. The

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 01 Issue 05
June 2014

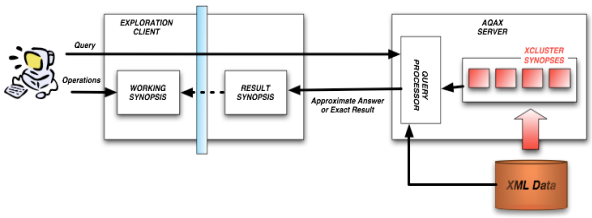
selection of our feature terms is not limited to the labels of XML elements.

- Each combination of the feature terms and the original query keywords may represent one of diversified contexts (also denoted as specific search intentions). And then, we evaluate each derived search intention by measuring its relevance to the original keyword query and the novelty of its produced results.
- To efficiently compute diversified keyword search, we propose one baseline algorithm and two improved algorithms based on the observed properties of diversified keyword search results.

ADVANTAGES OF PROPOSED SYSTEM:

- Reduce the computational cost.
- Efficiently compute the new SLCA results
- We get that our proposed diversification algorithms can return qualified search intentions and results to users in a short time.

SYSTEM ARCHITECTURE:



INTRODUCTION

MINING software repositories is an interdisciplinary domain, which aims to employ data mining to deal with software engineering problems [22]. In modern software development, software repositories are large-scale databases for storing the output of software development, e.g., source code, bugs, emails, and specifications. Traditional software analysis is not completely suitable for the large-scale and complex data in software repositories [58].

Data mining has emerged as a promising means to handle software data (e.g., [7], [32]). By leveraging data mining techniques, mining software repositories can uncover interesting information in software repositories and solve real world software problems.

A bug repository (a typical software repository, for storing details of bugs), plays an important role in managing software bugs. Software bugs are inevitable and fixing bugs is expensive in software development. Software companies spend over 45 percent of cost in fixing bugs [39].

Large software projects deploy bug repositories (also called bug tracking systems) to support information collection and to assist developers to handle bugs [9], [14]. In a bug repository, a bug is maintained as a bug report, which records the textual description of reproducing the bug and updates according to the status of bug fixing [64]. A bug repository provides a data platform to support many types of tasks on bugs, e.g., fault prediction [7], [49], bug localization [2], and reopenedbug analysis [63]. In this paper, bug reports in a bug repository are called bug data.

The primary contributions of this paper are as follows: 1) We present the problem of data reduction for bug triage. This problem aims to augment the data set of bug triage in two aspects, namely a) to simultaneously reduce the scales of the bug dimension and the word dimension and b) to improve the accuracy of bug triage.

2) We propose a combination approach to addressing the problem of data reduction. This can be viewed as an application of instance selection and feature selection in bug repositories. 3) We build a binary classifier to predict the order of applying instance selection and feature selection. To our knowledge, the order of applying instance selection and feature selection has not been investigated in related domains 2 BACKGROUND AND MOTIVATION

2.1 Background

Bug repositories are widely used for maintaining software bugs, e.g., a popular and open source bug repository, Bugzilla [5]. Once a software bug is found, a reporter (typically a developer, a tester, or an end user) records this bug to the bug repository. A recorded bug is called a bug report, which has multiple items for detailing the information of reproducing the bug. In Fig. 1, we show a part of bug report for bug 284541 in Eclipse.2 In a bug report, the summary and the description are two key items about the information of the bug, which are recorded in natural languages. As their names suggest, the summary denotes a general statement for identifying a bug while the description gives the details for reproducing the bug

DISCUSSION

In this paper, we propose the problem of data reduction for bug triage to reduce the scales of data sets and to improve the quality of bug reports. We use techniques of instance selection and feature selection to reduce noise and redundancy in bug data sets. However, not all the noise and redundancy are removed. For example, as mentioned in Section 5.2.4, only less than 50

percent of duplicate bug reports can be removed in data reduction (198=532 ¼ 37:2% by CH ! ICF and 262=532 ¼ 49:2% by ICF ! CH). The reason for this fact is that it is hard to exactly detect noise and redundancy in real-world applications. On one hand, due to the large scales of bug repositories, there exist no adequate labels to mark whether a bug report or a word belongs to noise or redundancy; on the other hand, since all the bug reports in a bug repository are recorded in natural languages, even noisy and redundant data may contain useful information for bug fixing

CONCLUSIONS

Bug triage is an expensive step of software maintenance in both labor cost and time cost. In this paper, we combine feature selection with instance selection to reduce the scale of bug data sets as well as improve the data quality. To determine the order of applying instance selection and feature selection for a new bug data set, we extract attributes of each bug data set and train a predictive model based on historical data sets. We empirically invest gate the data reduction for bug triage in bug repositories of two large open source projects, namely Eclipse and Mozilla. Our work provides an approach to leveraging techniques on data processing to form reduced and high-quality bug data in software development and maintenance. In future work, we plan on improving the results of data reduction in bug triage to explore how to prepare a highquality bug data set and tackle a domain-specific software task. For predicting reduction orders, we plan to pay efforts to find out the potential relationship between the attributes of bug data sets and the reduction orders.

, "Software fault prediction using quad tree-based k-means clustering algorithm," IEEE Trans. Knowl. Data Eng., vol. 24, no. 6, pp. 1146–1150, Jun. 2012.

[8] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," Data Mining Knowl. Discovery, vol. 6, no. 2, pp. 153–172, Apr. 2002.

[9] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Information needs in bug reports: Improving cooperation between developers and users," in Proc. ACM Conf. Comput. Supported Cooperative Work, Feb. 2010, pp. 301–310.

[10] V. Bol_on-Canedo, N. S_anchez-Maro~no, and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," Knowl.

Inform. Syst., vol. 34, no. 3, pp. 483–519, 2013.

[11] V. Cerver_on and F. J. Ferri, "Another move toward the minimum consistent subset: A tabu search approach to the condensed nearest neighbor rule," IEEE Trans. Syst., Man, Cybern., Part B, Cybern., vol. 31, no. 3, pp. 408–413, Jun. 2001.

[12] D. _Cubrani_c and G. C. Murphy, "Automatic bug triage using text categorization," in Proc. 16th Int. Conf. Softw. Eng. Knowl. Eng., Jun. 2004, pp. 92–97.

[13] Eclipse. (2014). [Online]. Available: http://eclipse.org/

[14] B. Fitzgerald, "The transformation of open source software," MIS Quart., vol. 30, no. 3, pp. 587–598, Sep. 2006.

[15] A. K. Farahat, A. Ghodsi, M. S. Kamel, "Efficient greedy feature selection for unsupervised learning," Knowl. Inform. Syst., vol. 35, no. 2, pp. 285–310, May 2013.

[16] N. E. Fenton and S. L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, 2nd ed. Boston, MA, USA: PWS Publishing, 1998.

[17] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in Proc. 13th Int. Conf. Mach. Learn., Jul. 1996, pp. 148–156.

[18] Y. Fu, X. Zhu, and B. Li, "A survey on instance selection for active learning," Knowl. Inform. Syst., vol. 35, no. 2, pp. 249–283, 2013.

[19] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," J. Mach. Learn. Res., vol. 3, pp. 1157–1182, 2003.

[20] M. Grochowski and N. Jankowski, "Comparison of instance selection algorithms ii, results and comments," in Proc. 7th Int. Conf. Artif. Intell. Softw. Comput., Jun. 2004, pp. 580–585.