Clustering of Defect Reports Using Graph Partitioning Algorithms

Vasile Rus[1], Xiaofei Nan[2], Sajjan Shiva[3], Yixin Chen[4]
[1,3]Dept. of Computer Science, University of Memphis, Memphis, TN 38152
[2,4]Dept. of Computer and Information Science, Univeristy of Mississippi, University, MS 38677
[1]vrus@memphis.edu, [2]xnan@olemiss.edu, [3]sshiva@memphis.edu, [4]ychen@cs.olemiss.edu

Abstract

We present in this paper several solutions to the challenging task of clustering software defect reports. Clustering defect reports can be very useful for prioritizing the testing effort and to better understand the nature of software defects. Despite some challenges with the language used and semi-structured nature of defect reports, our experiments on data collected from the open source project Mozilla show extremely promising results for clustering software defect reports using natural language processing and graph partitioning techniques. We report results with three models for representing the textual information in the defect reports and three clustering algorithms: normalized cut, size regularized cut, and k-means. Our data collection method allowed us to quickly develop a proof-of-concept setup. Experiments showed that normalized cut achieved the best performance in terms of average cluster purity, accuracy, and normalized mutual information.

1 Introduction

We address in this paper the challenging task of clustering defect reports. Defect reports are detailed descriptions in natural language of defects, i.e. problems in a software product. The proper handling of defect reports throughout the testing process for various purposes, such as fixing bugs in the case of developers, could have a great impact on the quality of the released software product. The defect reports are currently created and analyzed manually by testers, developers, and other stakeholders. Manual analysis is tedious, error-prone, and time consuming, leading to a less efficient testing process.

Defect reports are filed by testers (or users) who discover the defects through testing (or use). Reports include many details: an *id* that uniquely identifies the defect, the *status* of the defect (e.g. new, verified, resolved), a *summary* field, and a *description* field. The description field is the richest source of information about the defect. The field describes details in plain natural language about the defect, including symptoms and steps to reproduce the defect. The summary field is a one-sentence description of the problem.

We propose here advanced methods for clustering defect reports that take advantage of the description and summary fields of the reports. We regard each defect report as a textual document and use a well-known technique in information retrieval (IR), called the *vectorial representation*[1], to represent documents. As clustering algorithms, we applied the following three algorithms: k-means [4, 18], normalized cut[15], and size regularized cut [2]. This work extends our previous work on clustering defect reports in which we only experimented with the k-means clustering algorithm [13].

Three models were used to represent defect reports, one based on the summary field alone, one based on the description field alone, and another based on the union of both. Our experimental data consists of defect reports collected from the open source Mozilla project (www.mozilla.org). However, the proposed methods are transferable to defect reports from other projects, e.g. Eclipse (www.eclipse.org). The clustering was evaluated based on reports describing the same underlying problem. That is, defect reports are in the same cluster if they describe the same underlying problem.

As a preview of our results, we found that the normalized cut clustering algorithm [15] proved to be by far the most successful. Furthermore, using the union of the summary field and description field for clustering is better than using either the description field alone or summary field alone.

2. Related Work

There are two major lines of previous research relevant to our work: research on defect clustering, and research on using natural language processing (NLP) and information retrieval (IR) to mine artifacts from software repositories.

Clustering is the unsupervised classification of data points (usually represented as vectors in a multidimensional space) into groups (clusters) based on similarity [19]. The clustering problem has been addressed in many contexts

and by researchers in many disciplines. While we are not aware of any particular work on clustering defect reports, there is published research related to clustering defects in the manufacturing of semiconductors [6] and integrated-circuits (IC; [16]). Karnowski et al. [6] showed that fuzzy logic can help better cluster defects on semiconductor wafer maps. Singh and Krishna [16] have shown that using clustering information in optimization testing can significantly improve the shipped product.

The usage of NLP applications to improve software development and testing has been around at least since 1990s [14, 10, 11, 3]. More recently, there has been renewed interest in applying natural language techniques to mine useful artifacts from the various repositories associated with software projects (see the yearly Workshop on Mining Software Repositories at http://msr.uwaterloo.ca).

We discuss next a series of research efforts that are directly related to our work on clustering defect reports. Linstead et al. [8] described a framework to automatically mine developer contributions and competencies from a given code base, and extract software functions in the form of topics. Weiss and his colleagues [17] used k nearest neighbor to search for similar historical bug reports and further predict the fixing efforts. In other related work, annotation graphs have been used to identify bug-introducing changes [7] and different classification approaches, including Bayesian model, support vector machine, classification trees, and k-nearest neighbor, were tried for classifying software maintenance requests by Lucca et al. [9]. The use of the vectorial representation [1] to address the task of duplicate defect report identification has been investigated by Runeson et al. [12]. In this paper, we use the vectorial representation for the clustering of software defect reports. The clustering uses spectral graph partitioning algorithms, which are described in Section 3.

3. Spectral Graph Clustering

In recent years, spectral clustering based on graph partitioning theories has emerged as one of the most effective data clustering tools. Normalized cut (Ncut; [15]) is a graph bipartition method that attempts to organize nodes into groups so that the within-group similarity is high and the between-group similarity is low. Another graph partitioning method is size regularized cut(SRcut; [2]) which enables users to incorporate prior knowledge of the size of clusters into the clustering process and also minimizes the similarity between two clusters and, at the same time, searching for a balanced partition. Unfortunately, normalized cut and size regularized cut are both NP-complete problems. For Ncut, Shi and Malik proposed an approximated solution by solving a generalized eigenvalue problem [15]. As for SRcut, Chen et al. [2] proposed a relaxed version of the optimization that finds the largest eigenvalue of an associated matrix and uses it to bipartition the graph. These two methods can be recursively applied to get more than two clusters. In this work, we tested two heuristics for the Ncut clustering and one heuristic for SRcut clustering. With the first heuristic, used with both Ncut and SRcut, the subgraph with the maximum number of nodes is recursively partitioned (random selection is used for tie breaking). With the second heuristic, the subgraph with the minimal cut value is bipartitioned.

4. Experiments and Results

In this section, we address in detail the issues of data representation, similarity measure, and evaluation metrics. We also present performance results on clustering defect reports collected from Mozilla's Bugzilla, its the defect database.

4.1 Defect Reports Representation

A first issue we must address is the logical view of the defect reports (see [1] for more information on logical view). We chose a representation in which we retain all the words (after applying some preprocessing steps) but no positional information. Furthermore, we experimented with three models models for representing reports: using words in the summary, description, and the union of both. The advantage of using only the summary would be its relative small size, usually less than 50 words, which leads to fast clustering.

Another important issue to address is the formalism used for the representation. We used the vector space model [1]. A key feature in the vector space model is the weighting scheme of the words. We used the TF-IDF scheme (TF - term frequency; IDF - inverted document frequency).

4.2 Evaluation Metrics

Purity, accuracy, and normalized mutual information are our evaluation metrics. The purity of a cluster is the ratio of the dominant class size in the cluster to the cluster size itself. A larger value means that the cluster is a "purer" subset of the dominant class. We assign the dominant class of the documents within a cluster as the label of that cluster. A document is correctly clustered if its cluster label is identical to its class label provided by ground truth. The percentage of correctly clustered documents among the corpus is the accuracy. Purity and accuracy tend to increase with the number of clusters. However, mutual information is a measure that avoids this drawback. If normalized, mutual information values near 1 indicate that the similar partitioning, while a value close to 0 implies significantly different partitions.

4.3 Clustering Experiments

In our experiments, we chose to cluster bugs based on the fact that they describe the same defect. We regard a bug and its duplicates as a cluster. The data used in our experiments comes from Mozilla's Bugzilla, where accurate duplicate information about defects, as entered by human experts (developers), is available.

To create our experimental data set, we started collecting 20 Bugs from the *Hot Bugs List* of Mozilla's Bugzilla which contains the most filed recent bugs. We chose the top 20 defect reports from the *Hot Bugs List* in terms of largest number of duplicates and retrieved about 50 duplicates for each. We automatically collected the *Description* and *Summary* data of these bugs and stored them locally in text files. The final data set contained 1003 data points in 50 clusters. Some defect reports out of the 1020 that we collected initially (20 original defects at 50 duplicates each) were dropped because the description field was empty or was simply redirecting the user to another bug, e.g. the field contained textual pointers such as *see bug #123*. As such, the size of the clusters varies from 46 reports to 51 reports, i.e. we have approximately balanced cluster sizes.

For each report, three vectorial representations were created based on the description field, summary field, and the union of the two fields. The vocabulary size/ dimensionality for the three representations are 4569, 991, and 5128, respectively.

We applied three clustering algorithms, Ncut, SRcut, and k-means, to data set. It should be noted that the data set is balanced. That is, each cluster contains approximately same number of instances (50) as explained above. For the Ncut algorithm, two heuristics were tested to iteratively divide the data set into 20 clusters. In the first heuristic, named largest first (LF), the largest subgraph was divided in each iteration. In the second heuristic, named best first (BF), the subgraph with the minimal Ncut value was divided in each step. The SRcut used the first heuristic to iteratively generate 20 clusters. The α parameter of the SRcut algorithm was chosen to be 0.8.

The three evaluation metrics, average purity, accuracy, and normalized mutual information are reported in Table 1. Because the results of k-means depend on the initial choice of the seeds, we repeated 20 runs of k-means on the data set and reported the average and standard deviation of each metric. From Table 1, we can draw the following conclusions.

- Ncut with the largest first heuristic outperforms, on any evaluation metric given above, Ncut with the best first heuristic, SRcut and k-means algorithms on all three vectorial representations. The only exception is the average purity metric for the summaries data. This proves that the largest first heuristic Ncut method is suitable for balanced scenarios.
- The purity metric is biased towards smaller clusters. This is demonstrated by the k-means results. On all three vectorial representations, the average purity of k-means clustering is comparable to or higher than Ncut clustering. However, the accuracy and normalized mutual information of k-means are significantly lower than those of Ncut on all three vectorial representations. This is because k-means tends to generate a large number of small clusters.
- Using the combination of descriptions and summaries for clustering is better than using either representation separately.
- SRcut performed poorly on this data set. This is mainly due to the fact that the SRcut algorithm is designed for graph bipartition. When applied iteratively, it is difficult to find a proper value of the α parameter that works in all iterations.

5. Conclusions and Future Work

We addressed in this paper the challenging task of clustering defect reports based on their textual descriptions, summaries, and both descriptions and summaries. Our experiments on defect reports from Mozilla's Bugzilla and with three clustering algorithms showed that normalized cut using a TF-IDF vectorial representation based on a combination of descriptions and summaries of reports leads to better clustering than using the summary or the description of defects alone. Our work has been motivated by our belief that the rich information in software defect reports, which are generated during the testing phase in the form of textual reports, can be of great value. For instance, if open defect reports are clustered and a resulting cluster seems to be large compared to the others then the testing effort should focus, during the next testing cycles, on the defects in the large cluster. The large cluster may be an indication of an extremely faulty component or connected components which generate many related defects.

We plan to continue our investigation of clustering defect reports by using other representations of the defect reports, e.g. using only the *overview* section of the description field of a software report, and other text and knowledge processing techniques, e.g. exploiting knowledge about the particular software product being developed. As each defect report has a specific structure, we are also interested in exploring clustering techniques that take into account the structure instead of treating a report as a bag of words.

ACKNOWLEDGEMENT

The work of Rus and Shiva was sponsored by The University of Memphis under a Systems Testing Excellence

Table 1. Comparisons of clustering performance for Ncut, SRcut, and k-means algorithms with balanced class-size. Three vectorial representations (VR) are used: description field (VR1), summary field (VR2), and the union of description and summary fields (VR3). Ncut$_{LF}$ and Ncut$_{BF}$ denote Ncut algorithm with largest first heuristic and best first heuristic, respectively.

Evaluation Metric		Average Purity	Accuracy	Normalized Mutual Information
VR1	Ncut$_{LF}$	$\mathbf{0.8509}$	$\mathbf{0.8235}$	$\mathbf{0.8225}$
	Ncut$_{BF}$	0.8554	0.7468	0.7778
	SRcut$_{LF}$	0.6448	0.6471	0.6479
	k-means	$\mathbf{0.8566 \pm 0.0240}$	0.7579 ± 0.0382	0.7710 ± 0.0247
VR2	Ncut$_{LF}$	0.7856	$\mathbf{0.7677}$	$\mathbf{0.7525}$
	Ncut$_{BF}$	$\mathbf{0.8207}$	0.7069	0.7124
	SRcut$_{LF}$	0.6215	0.6092	0.5938
	k-means	0.7934 ± 0.0279	0.6252 ± 0.0250	0.6368 ± 0.0227
VR3	Ncut$_{LF}$	$\mathbf{0.8864}$	$\mathbf{0.8614}$	$\mathbf{0.8540}$
	Ncut$_{BF}$	0.8665	0.7587	0.7906
	SRcut$_{LF}$	0.6767	0.6800	0.6845
	k-means	$\mathbf{0.8879 \pm 0.0232}$	0.7899 ± 0.0354	0.8187 ± 0.0201

Program (STEP) project. The work of Nan and Chen was supported by The University of Mississippi.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[2] Y. Chen, Y. Zhang, and X. Ji. Size regularized cut for data clustering. In *Advances in Neural Information Processing Systems (NIPS)*, 18, MIT Press, Cambridge, pages 211–218, 2006.

[3] L. Etzkorn, L. Bowen, and C. Davis. An approach to program understanding by natural language understanding. *Natural Language Engineering*, 5(1):1–18, 1999.

[4] J. Hartigan and M. Wong. A k-means clustering algorithm. *Applied Statistics*, 28, 100–108, 1979.

[5] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[6] T. P. Karnowski, S. S. Gleason, and K. W. Tobin. Fuzzy logic connectivity in semiconductor defect clustering. In *Proc. SPIE Machine Vision Applications in Industrial Inspection VI*, A. R. Rao; Ning Chang; Eds., 3306, pages 44–53, 1998.

[7] S. Kim, T. Zimmermann, K. Pan, and E.J. Whitehead, Jr. Automatic idenfification of bug-introducing changes. In *Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering*, pages 81–90, 2006.

[8] E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi. Mining Eclipse developer contributions via author-topic models. In *International Workshop on Mining Software Repositories*, 2007.

[9] G. A. Di Lucca, M. Di Penta, S. Gradara. An approach to classify software maintenance requests. In *Proceedings of the International Conference on Software Maintenance (ICSM'02)*, pages 93–102, 2002.

[10] P. Lutsky. Documentation parser to extract software test conditions. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 294–296 1992.

[11] P. Lutsky. Using a document parser to automate software testing. In *Proceedings of the 1994 ACM Symposium on Applied Computing*, pages 59–63, Phoenix, Arizona, 1994.

[12] P. Runeson, M. Alexandersson, and O. Nyholm. Detection of duplicate defect reports using natural language processing. In *Proceedings of the 29th International Conference on Software Engineering*, pages 499–510, 2007.

[13] V. Rus, S. Mohammed, and S. Shiva. Automatic Clustering of Defect Reports. *Proceedings of the 20th International Conference on Software and Knowledge Engineering*, pages 291–297, 2008.

[14] J. Schlimmer. Learning meta knowledge for database checking. In *Proceedings of the National Conference of the American Association of Artificial Intelligence (AAAI'91)*, pages 335–340, 1991.

[15] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[16] A. Singh and C. Krishna. On the effect of defect clustering on test transparency and IC test optimization. *IEEE Transactions on Computers*, 45(6):753–757, 1996.

[17] C. Weiss, R. Premraj, T. Zimmermann, and A. Zeller. How long will it take to fix this bug?. In *Proceedings of the Fourth International Workshop on Mining Software Repositories*, pages 1–8, 2007.

[18] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, 2005.

[19] R. Xu and D. Wunsch. *Clustering.* John Wiley & Sons, 2008.