International Journal of Core Engineering & Management

ISSN: 2348 9510

International Journal Of Core Engineering & Management (IJCEM)
Volume 3, Issue 7, October 2016

AN APPROACH FOR GROUPING AND CLASSIFICATION OF BUGS FOR SOFTWARE ENGINEERING PRACTICES

Karthik B S
M.Tech Student, Dept. of CSE, Kuppam Engineering College
Kuppam, A.P, India
bashettykarthik@gmail.com

Thulasi Krishna S
Assoc. Prof, Dept. of CSE, Kuppam Engineering College
Kuppam, A.P, India
thulasikrishna1988@gmail.com

Abstract
In today's world number of software products coming into market according to growing demands and to serve human community. Each of the products has a list of known bugs or bugs created post release. In order to improve the quality of the product and to have early release of few bugs to the end consumer the product engineering, application development and Quality Assurance team sit together on a call, go through each and every bug one by one and then assign a priority and take those bugs into the development lifecycle of the product. In this paper algorithm is presented which performs the duplicate bug detection using a series of data mining techniques like Data Cleaning, Tokenization, Weight Computation, IDFT Computation, Score Computation and Duplicate Bug Detection. Classification of bugs is also performed for various sets of categories by using contingency and enhanced contingency algorithm. The Results show that the number of bugs to be discussed will get reduced in an automated fashion and also duplicate bugs are grouped.
Keywords- Software Quality, Data Mining, Software Products, Software Engineering, Duplication, Tokenization, Classification, Contingency.

I. INTRODUCTION

There are many works available in the literature related to the bug triage process. In the paper [1] software is used in which the testing team and the development team can report bugs and perform various product development related activities. The argument is made that the bug matching can be used by comparing the words and then if words of the 2 bugs are greater than 75% percent the bugs are treated as similar.

In the paper [2] the most common errors like script errors are described and a way to generate a test case by using automation frameworks is described. The algorithm does require the manual tester to write a script and then run an automation test case to group the bugs which is a very tedious process.

ISSN: 2348 9510
International Journal Of Core Engineering & Management (IJCEM)
Volume 3, Issue 7, October 2016

In the paper [3] first a bug repository is created and then machine learning algorithms are applied to classify the bugs and also architecture is developed to assign bugs to developers. However the algorithm suffers from accuracy and bugs are at random assigned to developer which increases the fix time drastically because the bug might be assigned to a developer who has little or no knowledge of the specific task.

In the paper [4] the vector space model is used for huge text data representation. It does not maintain the ordering of the words therefore authors produces an approach in which distance between the words in the graphs are used to intercept the information in terms of sentence structure of the underlying data.

II. PROPOSED FRAMEWORK

In the current approach it is assumed that we have a list of bugs across various products collected from open repository. The bugs undergo a series of data mining steps like cleaning, token formation, weight computation. Using the duplicate bug detection algorithm, duplicate bugs are eliminated there by using probability, contingency and enhanced contingency similar bugs are categorized based on various categories.

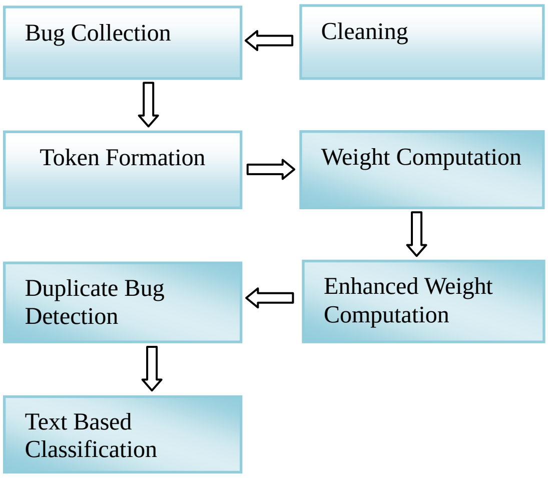


Fig. 1 Methodology for Duplicate Bug Detection

Figure shows the methodology for duplicate bug detection. As shown in figure series of steps are used for detecting duplicate bugs to improve quality.

A. Bug Collection

The bugs for all the above products namely Software Engineering for any of product, Mozilla, open office and eclipse. All the bugs will be collected as a set {Bug Id, Component, Priority, Type, Version, Status, and Description}.

B. Cleaning

This module is used in order to remove stop words from the bug description. The stop words used in this project are standard words given in the web mining forums. The stop words are namely able, about, above, abroad, according, accordingly, across, actually, adj, after, afterwards, again, against, Ago, ahead, ain't, all, allow, allows, almost, alone, along, alongside, already, also, although, always, am, Amid, amidst, among,

ISSN: 2348 9510
International Journal Of Core Engineering & Management (IJCEM)
Volume 3, Issue 7, October 2016

amongst, an and, another, any, anybody, anyhow, anyone, anything, anyway, anyways, anywhere, apart, appear, appreciate, appropriate, are, aren't, around, as, a's, aside, etc.,

C. Token Formation

Token Formation is a process of converting the clean bug into a sequence of tokens. Each token is associated with a bug {TokenId, TokenName, BugId, and ProductId}.

D. Weight Computation

It is defined as the number of times a token appears in the review. The Weight will remove if any redundancy exists. The Weight is stored in the format {FreqId, TokenName, Freq, BugId, ProductId}.

E. Score Computation

The Score computation is performed per token and is computed across the bugs by using the below formula and is stored in the format {ScoreId, Weight, IDFT, Score, BugId, ProductId}.

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \frac{f(q_i, D).(k_1+1)}{f(q_i, D) + k_1 (1 - b + b).\frac{|D|}{avgdl}}$$

$f = frequency$

$IDF = Inverse\ Document\ Frequency$

$D = length\ of\ document$

$avgdl = \text{average document length in the text collection}$

$\mathrm{k1} = 1.2$

$b = 0.75\ IDF(qi)$

Inverse Document frequency (IDF) can be computed by using the below formulae.

$$IDF(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

$N = number\ of\ documents$

$n(q_i) = number\ of\ documents\ containing\ q_i$

The similarity between 2 bugs d1 and d2 are measured using BM25F algorithm.

$$textual_{description}(d_1, d_2) = BM25F(d_1, d_2)$$

$Where,$

$d_1 = document1$

$d_2 = document2$

F. Duplicate Bug Detection

The algorithm is used to detect the whether the two bugs are similar or not. The algorithm finds the inter sum and union sum and then the bugs are found in terms of grouping.

International Journal of Core Engineering & Management
ISSN: 2348 9510
International Journal Of Core Engineering & Management (IJCEM)
Volume 3, Issue 7, October 2016

1. Consider the two bugs to be compared
2. Find the List of Concept words in Bug A1
3. Find the List of Concept words in Bug A2
4. Find the inter set between List of Concept Words in Bugs A1 and List of Concept Words in Bugs A2.
5. Find the union set between List of Concept Words in Bugs A1 and List of Concept Words in Bugs A2.
6. Start from index1 till the end of Concept Words in the inter set
   a. Obtain the kth Concept phrase K
   b. Measure the text Weight of Bugs A1 for K
   c. Measure the text Weight of Bugs A2 for K
   d. If tf(k,A1)>=tf(k,A2) measure the inter sum as below
      intersum = intersum+tf (k, A1)
      else
      intersum = intersum+tf (k, A2)
   e. k=k+1
   f. Repeat the process from step a to step e until all tokens in the inter set is exhausted
7. Start from index1 till the end of Concept Words in the union set.
   a. Obtain the k^{th} Concept phrase K
   b. Measure the text Weight of Bugs A1 for K
   c. Measure the text Weight of Bugs A2 for K
   d. If tf(k,A1)<tf(k,A2) measure the inter sum as below
      Uncommon sum = Uncommon sum +tf(k, A1)
      Else
      Uncommon sum = Uncommon sum +tf (k, A2)
   e. k=k+1
   f. Repeat the process from step a to step e until all tokens in the union set are exhausted
   g. Measure Similarity = InterSum/Uncommon sum

G. Classification

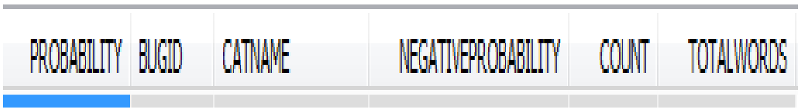
The algorithm is used post duplicate bug detection for classifying the bugs into various categories by computing the probability there by measuring the contingency and sorting according to category ratio.

1. Obtains the bugs from the collection
2. For each of the bugs the probability is computed using the following formula

   $$P(b \mid C_i) = \frac{Number\ of\ words\ of\ category\ C_i}{Total\ Number\ of\ Words}$$

   $1 \le C_i \le 4$
3. Also the negative probability is also computed for each of the bug
4. The probability computation is computed and constructed as below

iJCEM
International Journal of Core Engineering & Management
ISSN: 2348 9510
International Journal Of Core Engineering & Management (IJCEM)
Volume 3, Issue 7, October 2016



Probability- Positive Probability
BugID – ID of the Bug
CatName – c1, c2, c3 and c4
NegativeProbability – Finding the negative probability
Count- Number of words for the category
TotalWords- Number of words

5. The contingency is measure using the following

   $$Total\ +ve\ Other_{c1} = p(C2) + p(c3) + p(c4)$$

   $$Total\ \text{-}ve\ Other\ _{c1} = p^{1}(c2) + p^{1}(c3) + p^{1}(c4)$$

6. The enhanced contingency is measured using the following equation

   $$+ve\ Cat\ Ratio\ _{c1} = p(c1) + Total\ -ve\ Other\ _{c1}$$

   $$Other\ Cat\ Ratio_{c1} = p^{1}(c1) + Total\ +ve\ Other\ _{c1}$$

7. The bugs are then classified by order by positive category ratio maximum and other category ratio minimum

8. The count for each category bugs are then made

III. EXPERIMENTAL RESULTS

Web based software is used in which the developer first registers by giving his/her preferences and type of work. Two types of users are involved; Admin is responsible for sequence of algorithm operations as described in the methodology where as developers receive a bug based on their expertise. Whereas the developer can view the bugs assigned by the Admin based on the developer expertise.

A. Login Page

Fig. 2 Login Page

Figure shows the Login Page through which admin logs into the system and executes the algorithm.

iJCEM
International Journal of Core Engineering & Management
ISSN: 2348 9510
International Journal Of Core Engineering & Management (IJCEM)
Volume 3, Issue 7, October 2016

B. Bug Collection View

Bugs Information

Bug ID	Bug Details
1	Firefox choses wrong font for generic family with non-default font prefs
2	Add StringBuffer::finishAtom to create an atom from a string buffer
3	Google Map Maker is missing elements
4	Firefox doesn't pass iframe test from bug 363109 correctly
5	[css3-images] Radial gradients show the wrong color when there are 2 100% color stops ...
6	Firefox doesn't pass iframe from bug 363109 correctly

Fig. 3 Bug Collection View

Figure shows the list of bugs for the Firefox browser which admin can view after login.
Here in this paper we are taking a sample of bugs from various products like Mozilla, Eclipse, etc.

C. Data Cleaning Algorithm Output

Bugs Information

Bug ID	Bug Details
1	firefox choses wrong font generic family default font prefs
2	add stringbuffer finishatom create atom string buffer
3	google map maker missing elements
4	firefox doesn t pass iframe test bug correctly
5	css images radial gradients wrong color color stops end
6	firefox doesn t pass iframe bug correctly

Fig. 4 Data Cleaning Output

Figure shows the data cleaning output. As shown in the result all stop words are removed from the bug details. Hence the bug description will be free of stop words which will be more reliable and accurate for eliminating the duplicate bug's based on the concept words.

iJCEM
International Journal of Core
Engineering & Management
ISSN: 2348 9510
International Journal Of Core Engineering & Management (IJCEM)
Volume 3, Issue 7, October 2016

D. Tokenization and Weight Output

Bug ID	Product ID	Token Name	Frequency
1	1	firefox	1
1	1	choses	1
1	1	wrong	1
1	1	font	2
1	1	generic	1
1	1	family	1
1	1	default	1
1	1	prefs	1
2	1	add	1
2	1	stringbuffer	1
2	1	finishatom	1
2	1	create	1
2	1	atom	1
2	1	string	1
2	1	buffer	1
3	1	google	1
3	1	map	1

Fig. 5 Tokenization & Weight Output

Figure shows the Weight output as show in the matrix unique tokens are shown and then Weight is also shown.

E. Score Computation Output

Token Name	IDF	NVALUE	Score
firefox	0	6	40.1387189782676
choses	0.564271430438563	6	30.2480906732828
wrong	0.255272505103306	6	33.2450443999954
font	0.564271430438563	6	34.4984981468701
generic	0.564271430438563	6	30.2480906732828
family	0.564271430438563	6	30.2480906732828
default	0.564271430438563	6	30.2480906732828
prefs	0.564271430438563	6	30.2480906732828
add	0.564271430438563	6	31.2093832134072
stringbuffer	0.564271430438563	6	31.2093832134072
finishatom	0.564271430438563	6	31.2093832134072
create	0.564271430438563	6	31.2093832134072
atom	0.564271430438563	6	31.2093832134072
string	0.564271430438563	6	31.2093832134072
buffer	0.564271430438563	6	31.2093832134072
google	0.564271430438563	6	33.3277106103457

Fig. 6 Score Computation Output-1

Figure shows the Score computation output which contains the value computed for Inverse Document Frequency (IDF), N value and Score.

Token Name	Average D	Small N	B Value	Document Magnitude
firefox	7.16666666666667	3	0	8
choses	7.16666666666667	1	0.423203572828922	8
wrong	7.16666666666667	2	0.19145437882748	8
font	7.16666666666667	1	0.423203572828922	8
generic	7.16666666666667	1	0.423203572828922	8
family	7.16666666666667	1	0.423203572828922	8
default	7.16666666666667	1	0.423203572828922	8
prefs	7.16666666666667	1	0.423203572828922	8
add	7.16666666666667	1	0.423203572828922	7
stringbuffer	7.16666666666667	1	0.423203572828922	7
finishatom	7.16666666666667	1	0.423203572828922	7
create	7.16666666666667	1	0.423203572828922	7
atom				
string				
buffer				
google				

Fig. 7 Score Computation Output-2

ISSN: 2348 9510
International Journal Of Core Engineering & Management (IJCEM)
Volume 3, Issue 7, October 2016

Figure shows the values of the score formula computed token wise namely Average D, Small N, B Value and Document Magnitude

F. Duplicate Bug Detection

Main Bug ID	Bug ID	Group ID
1	1	1
1	2	0
1	3	0
1	4	0
1	5	0
1	6	0
2	2	2
2	3	0
2	4	0
2	5	0
2	6	0
3	3	3
3	4	0
3	5	0
3	6	0
4	4	4

Fig. 8 Duplicate Bug Detection

Figure shows duplicate bug detection which has 3 values Main Bug Id, Bug Id and Group Id. After apply the algorithm many bugs will belong to same group.

Union Sum	Intersection Sum	Similarity
1	1	1
477.588397385397	0	0
425.761267943276	0	0
488.841071964802	40.1387189782676	0.0821099561396054
475.109757078114	33.2450443999954	0.0699733985773092
462.969215367913	40.1387189782676	0.0866984621134477

Fig. 9 Duplicate Bug Parameters

Figure shows the computation of duplicate bug parameters. As shown in the fig there is union sum, inter sum and similarity if similarity is greater than threshold (0.6 or 0.7 or 0.8)

G. Classification Output

Once the duplicate bugs are detected using duplicate bug detection algorithm, only unique bugs are used for classification. The classification of bugs are done based on set of categories using text mining practices, results are described as below

Category Word	Category
quick response	Performance
extjs	Usability
javascript	Usability
jquery	User Interfacce
look and feel	Usability
database	Design
design	Design
load	Performance
resize	Usability
font	Usability
iframe	User Interface
Radial	Usability

Fig. 10 Category Words

Figure shows the category words. As shown in the fig there is category word and category to which the word belongs.

H. Probability Computation

The probability computation results are shown in the tabular format

Bug ID	Category Name	Count	Total Words	Probability	Negative Probability
1	Performance	0	9	0	1
1	Usability	2	9	0.22222222	0.7777777777777778
1	User Interfacce	0	9	0	1
1	Design	0	9	0	1
1	User Interface	0	9	0	1
2	Performance	0	7	0	1
2	Usability	0	7	0	1
2	User Interfacce	0	7	0	1
2	Design	0	7	0	1
2	User Interface	0	7	0	1

Fig. 11 Probability Computation

Figure shows the probability computation for the 2 bugs namely bug1 and bug2. The positive and negative probability for each category also has been computed.

I. Contingency

Contigency Information

Bug ID	Category Name	Negative Others	Positive Others
1	Performance	3	0
1	Usability	4	0
1	User Interfacce	3	0
1	Design	3	0
1	User Interface	3	0
2	Performance	4	0
2	Usability	4	0
2	User Interfacce	4	0
2	Design	4	0
2	User Interface	4	0

Fig. 12 Contingency

International Journal of Core Engineering & Management
ISSN: 2348 9510
International Journal Of Core Engineering & Management (IJCEM)
Volume 3, Issue 7, October 2016

Figure shows the contingency output. The positive others are the positive of other category and negative others is the probable weight of other categories.

J. Enhanced Contingency

Enhance Matrix

Bug ID	Category Name	Positive Cat Ratio	Others Cat Ratio
1	Performance	3	1
1	Usability	4	0.777777777777778
1	User Interfacce	3	1
1	Design	3	1
1	User Interface	3	1
2	Performance	4	1
2	Usability	4	1
2	User Interfacce	4	1
2	Design	4	1
2	User Interface	4	1

Fig. 13 Enhanced Contingency

Figure shows enhanced contingency which has the bug ids namely Bug1 and Bug2, Positive Category Ratio and Other Category Ratio are also computed for each category name.

K. Classifier Information

Classifier Information

Bug ID	Cat Name
1	Usability
2	Performance
2	Usability
2	User Interfacce
2	Design
2	User Interface

Fig. 14 Classifier Information

Figure shows the classified information as shown in the fig each bug belongs to either single category or multiple categories. Like this the output for huge number of bugs.

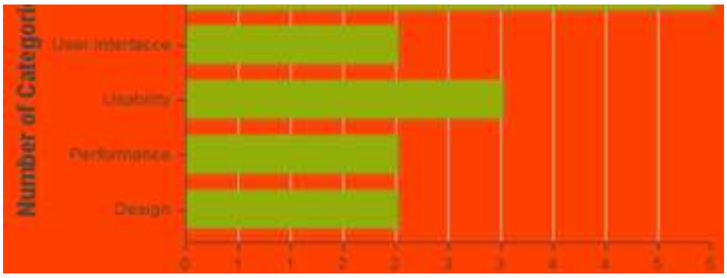


Fig. 15 Bug Classification

Figure shows the classification of bugs under various categories. As shown in the figure based on the classification algorithm 2 bugs belong to design, 2 bugs belong to performance, 3 bugs belong to usability and 2 bugs belong to user interface.

iJCEM
International Journal of Core Engineering & Management
ISSN: 2348 9510
International Journal Of Core Engineering & Management (IJCEM)
Volume 3, Issue 7, October 2016

L. Developer Registration

Fig. 16 Developer Registration

Figure shows the registration process used by the developer. The developer provides various fields namely First name, Last Name, Desired User Name, Password, Email Id and the category which the developer mostly works on.

M. Developers Bug Assignment

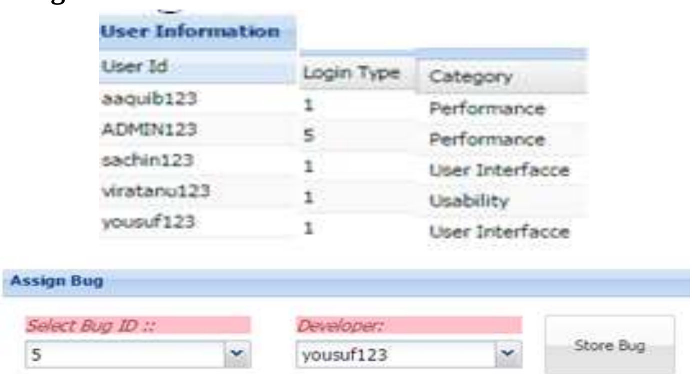


Fig. 17 Bug Assignment

Figure shows the Bug Assignment in which the bug id is being assigned to a developer. Here Bug Id is 5 and developer used is yousuf123.

N. Developers Bug

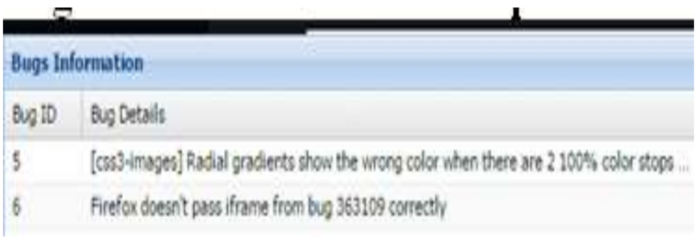


Fig. 18 Developers Bug

Figure shows the bug ids and details of the bugs assigned to the developers

ISSN: 2348 9510
International Journal Of Core Engineering & Management (IJCEM)
Volume 3, Issue 7, October 2016

IV. CONCLUSION

Duplicate Bug Detection is performed by doing a series of data mining operations where in duplicate bugs are eliminated.

The bugs are also classified into various categories by computing the probability, contingency and enhanced contingency and finally applying the classifier. This helps in assigning bugs to developer of that particular category.

V. FUTURE SCOPE

This work can be extended to support more products. The Classification can be also done graphically using k means along with applying the algorithm described in the paper for more accuracy.

REFERENCES

[1] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in Proc. 28th Int. Conf. Softw. Eng., May 2006, pp. 361–370.

[2] S. Artzi, A. Kie_zun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M. D. Ernst, "Finding bugs in web applications using dynamic test generation and explicit-state model checking," IEEE Softw., vol. 36,no. 4, pp. 474–494, Jul./Aug. 2010.

[3] J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions," ACM Trans. Soft. Eng. Methodol., vol. 20, no. 3, Bugs 10, Aug. 2011.

[4] C. C. Aggarwal and P. Zhao, "Towards graphical models for text processing," Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013.

[5] Bugzilla, (2014). [Online]. Avaialble: http://bugzilla.org/

Karthik B S received Bachelor's degree in Computer Science and Engineering from Jawaharlal Nehru Technological University Anantapur in 2014. Pursuing Master's in Computer Science and Engineering from Jawaharlal Nehru Technological University Anantapur.

Thulasi Krishna S received Bachelor's degree in Computer Science and Engineering from Jawaharlal Nehru Technological University Hyderabad in 2005, M.E from Sathyabama University, Chennai and pursuing PhD in Rayalaseema University, Kurnool. Working as Associate Professor in Kuppam Engineering College, Kuppam. Also a member of MIST and MIAENG.