ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Classification of Bug Reports Using Text Mining

[I]Suman, [II]Seema Rani, [III]Dr. Suresh Kumar
[I]M Tech. Scholar, [II]Assist. Professor, [III]Professor
[I,II,III]Dept. of CSE, Manav Rachna International University, Faridabad, Haryana, India

Abstract
All the organizations keep record of the bugs reported by various sources such as development teams, testing teams, and end users. Among these bugs some are security related and hence important to be resolved before resolving other common bugs. So, there is a need to classify these bugs as SBRs (security bug reports) or NSBRs (non-security bug reports). But, sometimes these reports are mislabeled by the bug reporter in a way that security faults (vulnerabilities) are labeled as non-security faults and probably may not be labeled at all. Correctly labeled bugs mitigate the vulnerabilities timely as well as appropriately. Due to lack of knowledge of bug reporter such kind of mislabeling could cause malfunctioning of software-system. To address this important issue, SAS text miner tool is used. Existing system SAS text miner classifies these bugs but for large dataset, it takes long time for classifying the reports. In this paper, we have proposed a new approach, in which we calculate the average length of the terms in the synonym list. After calculating the average, we have considered only those terms which are having length greater than or equal to the calculated average length. The terms having length less than the threshold value are not considered during classification This way we have reduced the number of terms which are to be matched with the synonym. So by ignoring the rest of the terms, we have saved a significant amount of time. Triager is the person who manually labels the bug.

General Terms
Start list, Stop list, Synonym list, Triager

Keywords
Bug Reports (BRs), SBR (Security Bug Report), NSBR (Non-Security Bug Report)

Introduction

Software is made up with many complexities and due to these intricacies generally software subsumes many bugs which cannot be detected at the time of development. These defects have caused billions of dollars lost [20]. One of the most frequent reasons for software maintenance phase which goes to 70 billion US dollars in the United States alone [21] is fixing defects. So many organizations maintain bug archives where a bug can be submitted by testers/users. Bugs arise from mistakes and errors, made by people, in either a program's source code or in the itself. If you want to detect and resolve the defect in early development stage, defect tracking and software development phases should start simultaneously. Bug fixing is an important activity for any software maintenance and its enhancement. In both open source and commercial projects, more than one person is involved in different capacities for resolving bugs. Software developers/testers report the bugs, explain the context of the observed bugs and help the team in fixing the bug. Bug triager prioritizes the bug and assigns it to appropriate people. Different people from the design team may suggest different solutions available for a bug fix, and help in choosing the best one to fix the bug. Thus effective bug reports leads to efficient bug fixing.

An analysis on bug reports shows that some bug reports are security related and some are general bugs, which do not have impact on the functionality of software and needs to be resolved after resolving the security related bugs. But, sometimes these reports are mislabeled by the bug reporter in a way that security faults (vulnerabilities) are labeled as non-security problems or vice versa and probably may not be labeled at all. Correctly labeled bugs mitigate the vulnerabilities timely as well as appropriately. Due to the large number of existing bug reports, it is a challenge for the triager to examine all existing bug reports manually and label them correctly. A triager is one who reports the bug. While reporting a bug a bug reporter may often mislabel SBRs as NSBRs due to lack of security domain knowledge. Hence it becomes necessary for the security engineers to examine NSBRs submitted to a BTS (Bug Tracking System) to recognize SBRs which are manually-mislabeled as NSBRs. However, manually scrutinizing (often thousands of) NSBRs in a BTS to identify SBRs is time consuming and often impracticable, or not even conducted in practice. Hence, there arises requirement of effective tool support for reducing human efforts in the process of identifying SBRs in a BTS, enabling this important security practice of SBR identification in either industrial or open source settings. With such effective tool support, security engineers make sure that the described security bug receives appropriate strengthening efforts, and gets fixed timely.

A SAS Text Miner, which is one of the existing approaches for identifying bug reports, smears text mining on natural-language descriptions of BRs to learn a statistical model to categorize a BR as either an SBR or an NSBR. Our approach classifies these bugs more efficiently and is able to handle much larger data set than the SAS Text Miner. The foundation of our proposed approach is to exploit valuable natural-language information of BRs in a BTS. Although bug reporters are not able to recognize that their bug is a security bug, we assume that the natural-language description of the bug in the BR may be sufficient to indicate that the bug is security-related and thus the BR is an SBR. To achieve this we manipulate valuable natural language information in BRs. We have evaluated our model on a large bug tracking system of company that contains thousands of docs and then applied the model on BRs that were labeled as NSBRs by bug reporters of the company. In this paper, we use the terms "bug reporters" and "security engineers" as follows. A bug reporter is any person who reports a bug to the company's BTS. A security engineer, who is liable for safeguarding software system. Security engineers assess SBRs that have been submitted to the BTS.

II. Text Mining Overview

Text mining is the exploration of data contained in natural language

text. It works by transposing words and phrases in unstructured data into numerical values which can then be linked with structured data in a database and can be analyzed with traditional data mining techniques. Text mining is the analysis of data contained in natural language text. The application of text mining techniques to solve business problems is called text analytics. Text mining can help an organization to potentially derive (derive potentially) valuable business insights from text-based content such as word documents and emails. Mining unstructured data is a challenge since natural language text is often inconsistent. It contains ambiguities caused by inconsistent syntax and semantics, including slang, language specific to industries and age groups. With an iterative approach, an organization can successfully use text analytics to gain insight into content-specific values such as sentiment, emotion, intensity and relevance. In text mining, the goal is to discover unknown information, something that no one yet knows and so could not have yet written down. Text mining is a variation on a field called data mining that tries to find interesting patterns from large databases. A typical example in data mining is using consumer purchasing patterns to predict which products to place close together on shelves, or to offer coupons for, and so on. The difference between regular data mining and text mining is that in text mining the patterns are extracted from natural language text where as in data mining patterns are extracted from structured databases of facts. Databases are designed for programs to process automatically whereas text is written for people to read. We do not have programs that can "read" text and may not have such probably for the future.

III. SAS Text Miner

Text documents that human being can easily understand must first are represented in a form that can be mined. The raw documents need to be processed first before patterns or concepts can be discovered. All this can be accomplished automatically with the help of SAS text miner tool. We have implemented the SAS Text Miner on a database of a company that contains thousands of bug reports having some mislabeled bug reports. This tool enables you to analyze the structured information that you have acquired from the text. This allows various options for parsing. It is possible to parse the document for detailed information about terms, phrases and other entities in the collection. This tool has extensive parsing capabilities that include:

- Stemming
- Automatic recognition of terms
- Normalization of various entities such as date, currency, percent and year
- Support for synonyms

1. Text Mining Process

1. File Processing: Creates a single SAS dataset from a set of documents. A dataset will be used as an input for the tool.

2. Text Parsing: Decomposes textual data and generates a quantitative representation suitable for data mining purposes.

3. Transformation: Transforms quantitative representation into a compact and informative format.

4. Document Analysis: Performs document classification of the set of documents.

Following are the key features of SAS:

1. Specify the input dataset
2. Text parsing and dimension reduction of the term by document frequency matrix.

2. Text Parsing

Text Parsing is the first step in analyzing a document collection. Parsing enables you to :

1. Break documents into terms.
2. Extract particular entities that are meaningful to your specific application.
3. Find the root form of word and specify synonyms.
4. Remove low information words such as a, an, the.
5. Identify the term's part of speech.
6. Creates a quantitative representation for the collection of documents.

3. Stop and Start List

All words in a document do not carry equally important information. For example, words such as prepositions, articles and conjunctions have little information in context of a sentence. Such low information words can be removed during text parsing since they often do not add useful information to the analysis. The stop list is a simple collection of low information or extraneous words that you want to ignore during processing.

The stop lists are useful for some, but not for all, text mining tasks. You can copy the stop word lists and edit the copy by removing or adding terms as necessary. One useful approach is to use the default stop word list and then examine the text parsing results. If you see some low informative words occurring in large percentage in documents, you might want to add these words to the stop word list and apply changes that are based on your judgment or domain knowledge. In other situations, some of the words in this stop list may not be informative for general text but within the given domain the words might have a specialized meaning. In this case, you can remove those words from the stop list so that they too would be kept in the analysis.

In contrast to the stop list, you can use a start list to control which words are included in your analysis. SAS text miner does not provide a default start list. The start list enables you to examine only certain words of interest. All other terms are removed from the results. This capability might be useful when you have previously mined data in this domain and have customized it.

4. Automatic Stemming

Stemming means finding and returning the root form of a word. Stemming enables you to work with linguistic forms that are more abstract than those of the original text. For example, the stem of corrects, corrected, correcting and corrective is 'correct'.

To better understand the advantage of analyzing more abstract terms, suppose that you are grouping documents according to the key words that they contain. But if this is done without stemming, words such as corrects, corrected, correcting and corrective will be handled unrelated words. Documents that contain one of these variants will not be treated the same as documents that contain the other variants.

5. Synonym

The document collection often contains terms that do not have the same base forms but share the same meaning in context. SAS uses a synonym list to store group of equivalent terms. The synonym list consists of records and each record contains the root term, an equivalent term, and the entity category. The way that SAS handles a term and its equivalent terms is equivalent to how stems and their roots are treated. A synonym list can be applied to other words that should be treated equivalently but are not direct stems.

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

For example, the words teach, instruct, educate and train do not have a common stem but share the same meaning of teach. In this case, you might want to add terms such as instruct, educate, teach and training into the synonym list in order for them to be also treated the same as teach.

6. Canonical Forms

Entities can also have synonyms. These are referred to as canonical forms. The text miner enables you to specify variant names of particular entities to be treated the same as their canonical form. Canonical form is standard name of a particular entity such as company name, date, currency and percentage expressions.

7. Customization

You can customize start, stop or synonym list. Depending on which dataset you are using, you can add or remove the terms.

8. Document Representation

One of the goals of text parsing is to create a computable representation for the documents. That is, in the process of text parsing, a term by document frequency matrix is created. Each entry in the matrix is represented by the number of times that a term appears in a document. Parsing the document collection generates the term by document frequency matrix. This term by document frequency matrix serves as the foundation for analysis of the document collection. Sometimes performance can be improved by adjusting the entries with various weighting functions. Often the function is based on how often a term occurs in the document collection as above. The terms that occur relatively frequently have the highest weight because the subsets of documents that include them are typically quite alike.

9. Weighting functions

The total weight of a term is determined by its frequency weight and term weight. Frequency weights are functions of the term frequency alone. Some of the frequency weights are binary, log etc. Term weights consider the word counts in the document collection. The following term weights are available –

A. Inverse document frequency

It uses the reciprocal of the number of documents that a term appears in the collection as the weight, for a similar result to the entropy method. This weighting method emphasizes terms that occur only in few document with in the collection. Term importance weighting are used to help distinguish some terms as being more important than others are. The general guideline is that only those terms are useful in categorizing a document which occurs frequently in that document, but rarely in other documents. The two weightings that perform similarity in this regard are entropy and inverse document frequency. The inverse document frequency is referred to tf-idf. For information retrieval and text mining research, using one of these two weighting gives the best performance. Term weighting has the effect of magnifying the importance of certain words while reducing importance of others. The complete matrix generally contains thousands of terms that are used throughout the document collection, of which only a small subset is contained in any one document. Therefore, computing with the entire set of terms is prohibitively expensive. Further, processing high dimensional data is inherently difficult for modeling. Reducing the dimensions will improve the performance.

IV. Existing Approach

Working of SAS text miner tool consists of three main steps. The first step is to obtain a bug report data set that contains textual descriptions of bugs. The BR data set is necessary for constructing and evaluating our natural-language predictive model. The second step is to create three database files which are used in text mining: a start list, a stop list, and a synonym list. The third step is to train, validate, and test the predictive model that calculates the probability that a BR is an SBR.

First, we obtain BRs which were submitted by development team, testing teams, and end users. Second, we distinguish between SBRs and NSBRs among the BRs. Then after obtaining the terms by doing parsing from the BRs, we select terms from them. Now the question is how to select the terms. We add the conjunctions, prepositions, articles and other words that have no useful information to the stop list. All terms in each bug report are checked if they are there in the stop list, then they are ignored and only the remaining terms take part in the stemming process. Stemming means finding and returning the root form of a word. Stemming enables you to work with linguistic forms that are more abstract than those of the original text. For example, the stem of corrected, corrects and correcting is correct. Thus we find more abstract terms and remove the redundant terms. Documents that contain one of these alternatives will not be treated the same as documents that contain the other alternatives. . To the synonym list, we add synonyms based on study of the enumerated terms from SBRs and NSBRs. Bug reporters may use security-related verbiage such as “run out of memory” or “memory overrun” to describe the same bug. By including such terms in the synonym list, the predictive model can classify different terms in the same milieu to reflect the same type of bugs. We next use SAS Text Miner to generate a term-by document frequency matrix from the terms in BRs based on the start or stop, and synonym lists.

To the start list, we manually add terms such as “vulnerability” and “attack” from SBRs. We also include terms (from SBRs) that are not explicitly security-related, but can indicate a security problem. For example, “destruct” and “failure” are also candidates for inclusion in the start list. To the stop list, we add prepositions, articles, and conjunctions since they likely have little benefit for indicating a security bug. Both stop lists and start lists are acceptable for text mining. SAS Text Miner allows either a start list or stop list to be used in text mining, but not both. To the synonym list, we add synonyms based on examinations of the enumerated terms from SBRs and NSBRs. Bug reporters may use security-related verbiage such as “buffer overflow” or “buffer overrun” to describe the same bug. By including such terms in the synonym list, the predictive model can identify different terms in the same context to reflect the same type of bugs. We next use SAS Text Miner to generate a term-by document frequency matrix from the terms in BRs based on the start or stop, and synonym lists. A term-by document frequency matrix basically contains all the parsed and stemmed terms. The documents having those terms and their frequency in the related documents are maintained as follows:

Table 1 : Term by document frequency matrix

Term	Document	Frequency
Failure	2,1	5
Buffer Overflow	1,3	3

After creating a matrix, calculate the weights of the terms in the related documents. These weights indicate the importance of the term in the corresponding documents. Weight can be calculated by using various functions, but term frequency inverse document frequency method gives the best results. The total weight of a term is determined by the frequency weight and the term weight. We use the log frequency weight function to lessen the effect of a single term being repeated often in each BR. We use the term frequency inverse document frequency term weight function. This weighting method emphasizes terms that occur only in few document with in the collection. Term importance weighting are used to help distinguish some terms as being more important than others. The general guideline is that only those terms are useful in categorizing a document which occurs frequently in that document, but rarely in other documents. The document that contains those terms will be easy to set apart from the rest of the collection.

The inverse document frequency is refers to that tfidf. In the information retrieval and text mining research, using one of these weighting functions gives the best performance.

How to Compute

Typically, the tf-idf weight is computed by using two terms: first, Normalized Term Frequency (TF) (the number of times a word appears in a document, divided by the total number of words in that document); second, Inverse Document Frequency (IDF) (computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears).

Term Frequency

It measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length as a way of normalization:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

Inverse Document Frequency

It measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

IDF (t) = loge (Total number of documents / Number of documents with term t in it).

TFIDF=TF*IDF

After computing the weights, match the terms which have the tf-idf value greater than threshold value with the synonym table. Now label the documents which have the identified matched terms as SBR and remaining as NSBR.

V. Purposed Approach

The data is collected from various sources such as end users, testers and development teams. When a bug is reported by a bug reporter to a Bug Tracking System, the reporter is required to label the bug report as Security Bug Report (SBR) or Not Security Bug Report (NSBR). It is necessary to prioritize the SBRs, to fix them before fixing NSBRs. But, what happen when reporters mislabel the SBRs as NSBRs? It may cause serious problems to our system if we do not fix SBR in specific time limit. So, to identifying NSBR as SBR we use text-mining of summary reports given in summary attribute with respect to considered attributes of product, component, resolution and state. So, with the help of stemming, we compact the terms by removing their different grammatical forms. These compressed data terms help in efficient analysis/classification. From this we will get a set of root words related to security, which will help in identifying whether a report is security related or not.

Now, after studying the working of SAS text miner we are able to classify the bug reports as SBR or NSBR. When we implemented the SAS, we found that the trained model is usable and we can feed new BR data set and BRs without labels to the model for predicting their labels. In our approach, we eradicate the use of stop word list. Instead of using stop word list, we have followed a new approach, in which, after loading the dataset, we first calculate the average length of the terms in the synonym list. After calculating the average, length we have considered only those terms which are having length greater than or equal to the calculated average length. This way we have reduced the number of terms which are to be matched with the synonym. So by ignoring the rest of the terms, we have saved a significant amount of time without affecting the system accuracy. This will significantly reduce the execution time. We made a distinction between the terms according to the average length. After finding the distinguished terms, we have again filtered the terms and group the terms on the basis of bug report. For example, BR1 and BR5 contain term attack, and then BR1 and BR5 both are grouped by the term 'attack'. After accomplishing this task, now again distinguishes the terms on the basis of whether they are related to security or non-security terms and matches the security related terms to the synonym table and labeled the reports as SBR which contains the security related terms.

After accomplishing this task, now again distinguishes the terms on the basis of whether they are related to security or non-security terms and matches the security related terms to the synonym table and labeled the reports as SBR which contains the security related terms.

We implemented this approach on the dataset of 1000 bug reports which were collected from various sources and achieved improved execution time over an existing approach. If we input the given dataset to an existing system then it takes long time in categorizing the documents and if number of records is increased then system becomes too slow or unresponsive. Existing approach works efficiently with small no of records, but as the number of records increases, execution time also increases or you can say, It gives good result with a certain limited number of records. But with our proposed approach we have overcome this problem and achieved the improved execution time no matter how many records are there. In existing approach, after removal of stop words and stemming, the classification (of SBR and NSBR) is done by term by term matching and this matching is done two times, first it matches the document with stop word table and then after stemming process, terms are again matched with synonym table. It is very time consuming and takes long time for classification or if number of records are more then there is a probability that the system may get halt. Instead of matching all the terms, we take the average length of terms of synonym table and match only those terms, which are having greater or equal length to the calculated average length. So by not considering rest of the terms, it does not affect much on correct labeling of bug reports. And by undertaking this approach, we shrink the execution time.

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

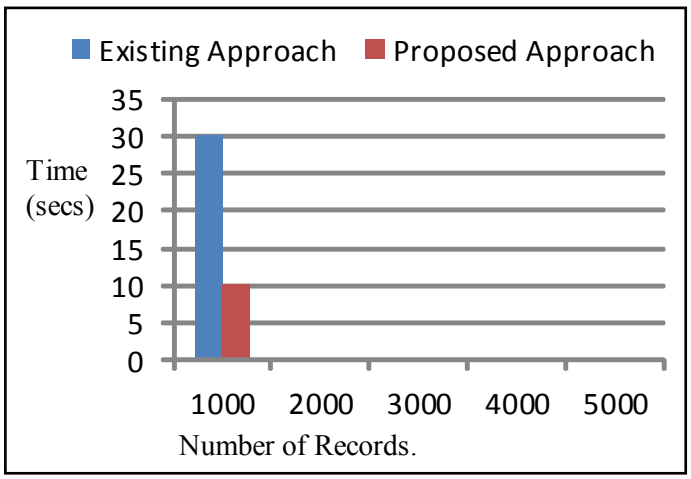

Fig. 1 : Performance

VI. Conclusion

We found that SAS Tex Miner which uses a stop list and a synonym list takes more execution time and is not feasible for the large data set. We chose not to use a stop list for our text mining because we are uncertain that continually updating the start list is more feasible for a limited number of security bugs than managing a large stop list. As an alternative of stop list, we take average of length of max and min length of terms in synonym list and then take only those terms whose length is greater or equal to the average length. Hence we are able to reduce the execution time and classify the bugs efficiently.

References

[1]. *Anvik, J., Hiew, L., and Murphy,"Coping with Open Bug Repositories", pp.35-39, 2005.*

[2]. *Anvik, J., Hiew, L., and Murphy" Who Should Fix This Bug?", pp.371-380, ICSE., 2006.*

[3]. *A. Ko and B. Myers.,"A linguistic analysis of how people describe software problems", In IEEE Symposium on Visual Languages and Human-Centric Computing, pages 127–134, 2006.*

[4]. *A. Podgurski, D. Leon, P. Francis, W. Masri, M. Minch, J. Sun, and B. Wang, "Automated Support for Classifying Software Failure Reports" Proc of the ICSE, pp. 465-475, 2003.*

[5]. *C. Anley, "Advanced SQL Injection In SQL Server Applications,"Next Generation Security Software Ltd, 2002.*

[6]. *C. Bird, A. Bachmann, E. Aune, J. Duffy, A.Bernstein, V. Filkov, and P. Devanbu, "Fair and balanced?: bias in bug-fix datasets", Proc. of the Joint Meeting of the European Softw. Eng.Conf. and the ACM SIGSOFT Symp. on the Foundations of Softw. Eng., pages 121–130, NewYork, NY, USA, 2009. ACM.*

[7]. *C. Fox." Lexical analysis and stoplists. In Information Retrieval - Data Structures & Algorithms", pages 102,130. Prentice-Hall, 1992.*

[8]. *Clause, J. and Orso, A," A Technique for Enabling and Supporting Debugging of Field Failures", pp.261-270, In Proc. ICSE, 2007.*

[9]. *Cubranic, D. and Murphy," Automatic Bug Triage Using Text Classification", pages 92-97, In Proc. SEKE, 2004.*

[10]. *D. Cubranic and G. Murphy, "Automatic Bug Triage Using Text Classification" Proc of the SEKE, pp. 92-97, 2004.*

[11]. *D. Hawking. Overview of the TREC2002. In Proceedings of the Nineth Text Retrieval Conference (TREC 9), pages 87-94, Gaithersburg, MD, 2000.*

[12]. *D. Hawking, E. Voorhees, N. Craswell, and P. Bailey. Overview of the TREC-8 Web Track. In Proceedings of the Eighth Text Retrieval Conference (TREC-8), pages 131-150, Gaithersburg, MD, 1999.*

[13]. *E. Voorhees and D. Harman. Overview of the Eighth Text Retrieval Conference (TREC-8). In Proceedings of the Eighth Text Retrieval Conference (TREC-8), pages 1-23, Gaithersburg, MD, 1999.*

[14]. *Elbaum S. and M. Diep. Profiling Deployed Software: Assessing Strategies and Testing Opportunities. IEEE TSE, 31, pages 312–327, 2005.*

[15]. *E. M. Voorhees. Overview of TREC2002. In Proceedings of the Eleventh Text Retrieval Conference (TREC2002), pages 1-16, Gaithersburg, MD, 2002.*

[16]. *Francis, P., Leon, D., and Minch, M. Tree-Based Methods for classifying Software Failures. In Proc. ISSRE, 2004, 451-462.*

[17]. *G. Amati. Probabilistic Models for Information Retrieval based on Divergence from Randomness. PhD thesis, Department of Computing Science, University of Glasgow, 2003.*

[18]. *G. Amati and C. J. van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Transactions on Information Systems (TOIS), 2002,pages 357-389.*

[19]. *G. Jeong, S. Kim, and T. Zimmermann, "Improving Bug Triage with Bug Tossing Graphs" Proc of the ESEC-FSE, 2009.*

[20]. *G. Tassey. The economic impacts of inadequate infrastructure for software testing. National Institute of Standards and Technology. Planning Report 2-3, 2002.*

[21]. *J. Sutherland. Business objects in corporate information systems. In ACM Computing Surveys, 2006.*

[22]. *N. Bettenburg, S. Just, A. Schroter, C. Weiss, R. Premraj, and T. Zimmermann, "What Makes a Good Bug Report?" Proc of the FSE, pp. 308-318, 2008.*

[23]. *N. Bettenburg, R. Premraj, T. Zimmermann, and S.Kim,"Duplicate Bug Reports considered Harmful?" Proc of the ICSM, pp. 337-345, 2008.*

[24]. *http://www.sas.com/technologies/analytics/datamining/textminer/*