How Stemming Help to Track Bugs in Bug Tracking System

Pankaj kumar[1] Mr.Archit Kumar[2]
[2]Assistant Professor
[1,2]Dept. of C.S.E
[1,2]CBS Group of Institutions,Haryana, India.

Abstract— In this paper we analyze theoretically how steaming help to track bugs in bug tracking system (BTS). Stemming is very important part of natural language processing (NLP). NLP is used to find bug related terms by reading line by line text summary that is given by Users, developer, and testers. The Bug Tracking System (BTS) is use to maintain bug related database with important attributes i.e. ID, Product, Component, Assignee, Status, Resolution, Summary, Changed Time and Category. The data is collected from various sources such as end users, testers and development teams. When bug is reported by bug reporters to a Bug Tracking System, the reporters required to label the bug reports as Security Bug Reports (SBR) or not i.e. Not Security Bug Reports (NSBR). It is necessary to prioritize the SBRs, to fix first then NSBRs. But, what happen when reporters misclassify the SBR as NSBR? It's may cause serious problems to our system if we not fix SBR in specific time. So, to identifying NSBR is 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 stemming we compact the terms by removing the grammatical form of the words. These compressed data term more understandable form. By this way we get best security related words. That help to find what actually our report is SBR or NSBR.

Key words: Bug Tracking System; Text Mining; Suffix; IR; NLP.

I. INTRODUCTION

The purpose of Bug Tracking for improving software reliability is to provide better service to the administrator or useful for applications developed in an organization. In a BTS, some BRs are labeled by bug reporters as security bug reports (SBRs), whose associated bugs are found to be security problems. SBRs generally deserve higher fix priority than not-security bug reports (NSBRs), the subset of BRs that are believed not to have a security impact. Correctly labeling SBRs among BRs submitted to a BTS is important in security practice since delay of identifying and fixing the security bugs involved in the SBRs causes serious damage to software-system stakeholders.

We have many stemming algorithm. Our aim to find good algorithm that help to guide the bug tracking system. In the stemming we considers the words which carry similar meanings but in different grammatically form (such as "load" and "loading") therefore it is needed to combine them into one term. In this way, the documents can show a better representation (with stronger correlations) of these terms and even the dataset can be reduced for achieving faster processing time.

II. CATEGORIES OF STEMMING ALGORITHMS

The stemming algorithm has been categorized into three main categories and they are statistical methods, Affix removal methods and other methods [1].

1) Statistical methods

These are the stemmers who are based on statistical analysis and techniques. Most of the methods remove the affixes but after implementing some statistical procedure

A. HMM Stemmer

This stemmer is based on the concept of the Hidden Markov Model (HMMs) which are finite-state automata where transitions between states are ruled by probability functions. At each transition, the new state emits a symbol with a given probability. This model was proposed by Melucci and Orio[2].

B. N-Gram Stemmer

This is a very interesting method and it is language independent. Over here string-similarity approach is used to convert word inflation to its stem. An n-gram is a string of n, usually adjacent, characters extracted from a section of continuous text. To be precise an n-gram is a set of n consecutive characters extracted from a word. The main idea behind this approach is that, similar words will have a high proportion of n-grams in common. For n equals to 2 or 3, the words extracted are called digrams or trigrams, respectively.

C. YASS Stemmer

The name is an acronym for Yet another Suffix Striper. This stemmer was proposed by PrasenjitMajumder,et. al. [3]. According to the authors the performance of a stemmer generated by clustering a lexicon without any linguistic input is comparable to that obtained using standard, rule-based stemmers such as Porter's.

2) Affix Removal methods

As the name clearly suggests these methods are related to removing the suffixes or prefixes (commonly known as affixes) of a word.

A. Lovins Stemmer

This was the first popular and effective stemmer proposed by Lovins in 1968. It performs a lookup on a table of 294 endings, 29 conditions and 35 transformation rules, which have been arranged on a longest match principle [4].

B. Porters Stemmer

Porters stemming algorithm [5,6] is as of now one of the most popular stemming methods proposed in 1980. Many modifications and enhancements have been done and suggested on the basic algorithm. It is based on the idea that the suffixes in the English language (approximately 1200) are mostly made up of a combination of smaller and simpler suffixes.

3) Other methods

A. Krovetz Stemmer (KSTEM)

The Krovetz stemmer was presented in 1993 by Robert Krovetz [7] and is a linguistic lexical validation stemmer. Since it is based on the inflectional property of words and the language syntax, it is very complicated in nature.

B. Context Sensitive Stemmer

This is a very interesting method of stemming unlike the usual method where stemming is done before indexing a document, over here for a Web Search, context sensitive analysis is done using statistical modeling on the query side. This method was proposed by Funchun Peng et. al.[8]. Basically for the words of the input query, the morphological variants which would be useful for the search are predicted before the query is submitted to the search engine. This dramatically reduces the number of bad expansions, which in turn reduces the cost of additional computation and improves the precision at the same time.

III. ERRORS IN STEMMING

There are mainly two errors in stemming – over stemming and under stemming. Over-stemming is when two words with different stems are stemmed to the same root. This is also known as a false positive. Under-stemming is when two words that should be stemmed to the same root are not. This is also known as a false negative. Paice has proved that light-stemming reduces the over-stemming errors but increases the under-stemming errors. On the other hand, heavy stemmers reduce the under-stemming errors while increasing the over-stemming errors [1 , 2].

IV. PROPOSED APPROACH

In proposed approach we use porters stemmers algorithm for our Bug tracking system. We use porters stemmers algorithm because it is Produces the best output as compared to other stemmers. It's making less error rate compared to others. Compared to Lovins it's a light stemmer. The Snowball stemmer framework designed by Porter is language independent approach to stemming. All these advantages make its good for our purpose.

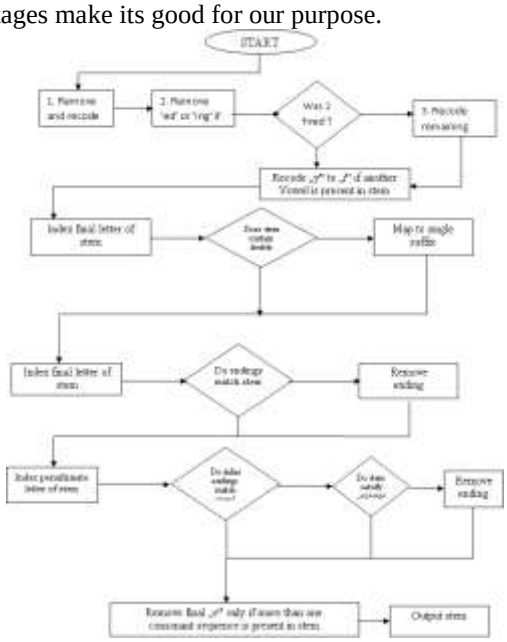

Fig . 1**:** Porter Stemming Algorithm

Porter Stemmer Steps:-

Step 1: Gets rid of plurals and -ed or -ing suffixes.

Step 2: Turns terminal y to i when there is another vowel in the stem.

Step 3: Maps double suffixes to single ones:- ization, -ational, etc.

Step 4: Deals with suffixes, -full, -ness etc.

Step 5: Takes off -ant, -ence, etc.

Step 6: Removes a final –e.

V. RESULT

Bug tracking dataset have different attributes such as product, component, status, resolution, Summary and its category stage. We fetch the summary for text mining in bug tracking system.

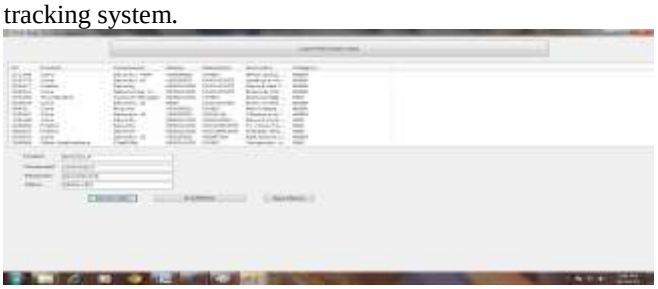

Fig. 2: Get data from BTS data base

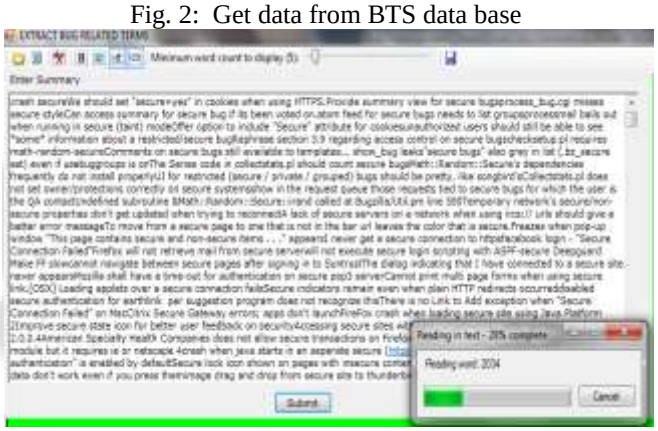

Fig . 3: Extract bug Related terms

Now we have to evaluate the bug related text. With the help of natural language processing we read line by line text and only consider security related terms like vulnerability, attack etc.

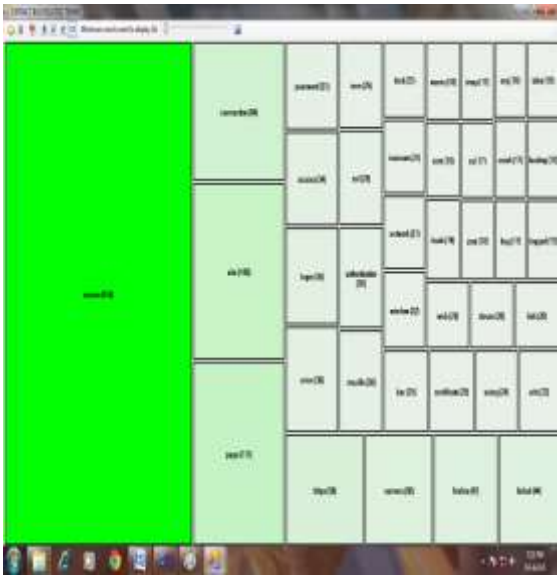

Fig . 4: Bug related terms

After processing we get the terms with the frequency, in grid forms. If the terms are security related terms then the bug report for corresponding attributes are SBR.

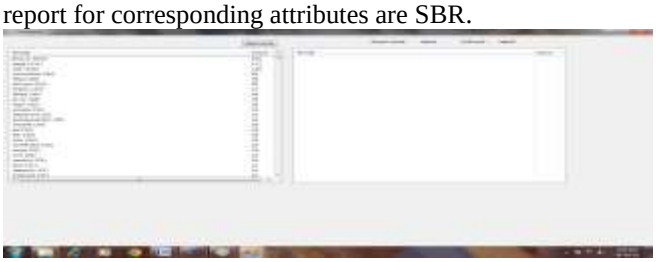

Fig . 5: Stemming

After natural language processing we start for the stemming of words. In the stemming we considers the words which carry similar meanings but in different grammatically form (such as "load" and "loading") therefore it is needed to combine them into one term. In this way, the documents can show a better representation (with stronger correlations) of these terms and even the dataset can be reduced for achieving faster processing time.

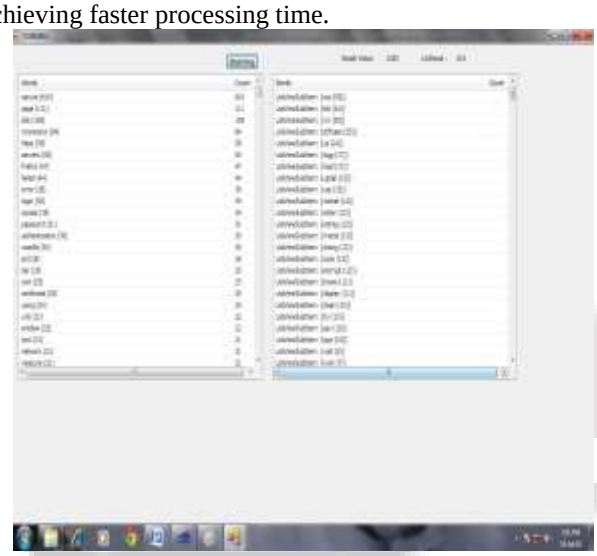

Fig . 6: After Stemming

From the stemming we get the total weight of a term. Weight is determined by the frequency of terms. Weight is indicated by counts column in above screen shot. Higher weighting functions indicate the categories of bug in above we get SBR categories bug.

Performance

The accuracy of this algorithm is tested by calculating mean number of words (MWC) by dividing the number of Words entered with the accurate words after stemming.[10].

MWC = W / A. (1) Where,

W = Number of words entered.

A = Accurate words after stemming.

To test the accuracy of Porter Stemming algorithm, we enter 1452 words as input and the accurate number after Stemming is 415. By substituting these values in (1), we got:

MWC = 1452 / 415

MWC = 3.499

VI. CONCLUSION

In this paper we implement a Porter Stemming algorithm in bug tracking system to extract bug related terms. That makes able to find SRB that are mislabeled by end users, testers and developers. By this way we enhance the search results of security related bugs. This would be a healthy change to the current situation where they all provide identical functionality. In future with the help of stemming algorithm we use bug tracking system as a expert system which analyze our bug very efficiently and provide solution for the bugs. In future we use bug tracking system as a expert system which analyze our bug very efficiently and provide solution for the bugs. Also in future we in-builds these system in developing software to makes a report about bug .

REFERENCES

[1] Anjali Ganesh Jivani et al, Int. J. Comp. Tech. Appl., Vol 2 (6), 1930-1938Paice Chris D. "A Comparative Study of Stemming Algorithms". ACM SIGIR Forum, Volume 24, No. 3. 1990, 56-61. ISSN:2229-60932011

[2] Paice Chris D. "An evaluation method for stemming algorithms". Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval. 1994, 42-50.

[3] J. B. Lovins, *"Development of a stemming algorithm,"* Mechanical Translation and Computer Linguistic., vol.11, no.1/2, pp. 22-31, 1968.

[4] Porter M.F. "An algorithm for suffix stripping". Program. 1980; 14, 130-137.

[5] Porter M.F. "Snowball: A language for stemming algorithms". 2001.

[6] Melucci Massimo and Orio Nicola. "A novel method for stemmer generation based on hidden Markov models". Proceedings of the twelfth international conference on Information and knowledge management. 2003, 131-138.

[7] Prasenjit Majumder, Mandar Mitra, Swapan K. Parui, Gobinda Kole, Pabitra Mitra and Kalyankumar Datta. "YASS: Yet another suffix stripper". ACM Transactions on Information Systems. Volume 25, Issue 4. 2007, Article No. 18.

[8] Noraida Haji Ali , et al. "Porter Stemming Algorithm for Semantic Checking" ICCIT 2012.

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

[10] 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.

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

[12] P. Cerrito, *Introduction to Data Mining*, Cary, SAS Institute, Inc., 2006.

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

[14] M. Howard, D. LeBlanc, and J. Viega, *19 Deadly Sins of Software Security: Programming Flaws and How to Fix Them*, Emeryville, McGraw-Hill/Osborne, 2005.

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

