

STATISTICAL PROBABILITY BUG TRACKING SYSTEM USING NAÏVE BYES IN DATA MINING

¹Er. Amandeep, ²Er. Saurabh Mittal

¹M. Tech. Scholar, CSE Department,

Galaxy Global Imperial Technical Campus, Dinarpur Ambala (India)

²Associate Professor, CSE Department,

Galaxy Global Group of Institutions, Dinarpur, Ambala, (India)

ABSTRACT

Bug-tracking mechanism is employed in software development houses to track the bugs in the software. We are aimed at distinguishing the very fast and the very slow bugs so as to prioritize them while working on them. We computed our prediction model using Naïve Bayes classifier. Simply relying on shared lists and email to monitor the status of defects is error-prone approach and the possibility is that the bug judged by developer to be insignificant is ignored.

Bug Tracking System or Defect Tracking System is an ideal solution for individual or groups of developers to track the bugs of a product, solution or an application.

The Bug Tracking System can dramatically increase the productivity and accountability of individual employees by providing a documented target based workflow and positive feedback for good performance.

Bug Tracking System allows below functionalities:

1. Creating & Changing Bugs at ease
2. Query Bug List to any depth
3. Reporting & Charting in more comprehensive way
4. Multi-level Priorities & Severities.
5. Attachments & Additional Comments for more information

I INTRODUCTION

In a Bug Tracking System, some Bug Reports are labeled as security bug reports (SBRs), whose associated bugs are found to be security problems. These SBRs must be fixed on priority than not-security bug reports (NSBRs), the subset of BRs that are believed not to have a security impact. Correctly labeling and fixing SBRs among BRs submitted to a BTS is important in security practice so that there is no delay causing serious damage to software-system stakeholders. The likelihood of unlabeled SBRs in a BTS could be high for at least three reasons.

- 1) If bug reporters perceive a subtle security bug that they are reporting in a BR as an innocuous not-security bug, then they may label the BR as an SBR.

- 2) Some security bugs described in BRs are associated with recommended mitigations that may be unknown to bug reporters.
- 3) Bug related to general reliability problems can also be related to security problems and a bug reporter without sufficient security knowledge may report this bug as a NSBR.

Therefore, there remains a strong need of effective tool support for reducing human efforts in this process of identifying SBRs in a BTS, enabling this important security practice of SBR identification in either industrial or open source settings.

II LITERATURE REVIEW

C. Code Search and Recommendation Using Stackoverflow Zagalsky et al. describe a code search and a recommendation tool called as Example Overflow which mines information present in Stack Overflow (Q&A website for programmers) [12]. Their work is motivated by the need to minimize context switch between development environment and code-search tools.

In context to existing work, the study presented in this paper makes the following novel contributions:

- 1) The work described in this paper is the first focused study on integration of issue tracking systems with community driven question and answering websites such as Stack overflow. While there has been work in the area of code-editors & development environment integration (Section II-A) with Stack overflow as well as code-editor integration with web-search and external websites (Section II-B), the integration of Stack overflow with issue tracking systems is a unique research direction.
- 2) We present experimental results (based on a series of experiments conducted on publicly available dataset from two popular, large, complex and open-source projects: Google Chromium and Android) indicating presence of several links to Stack overflow question & answers facilitating the process of bug resolution. We conduct a characterization study and present our perspective on the correlation between Stack overflow references and mean time to repair a bug, top domains in issue tracking system threaded discussion forums and illustrative examples showing links to various Web 2.0 platforms in addition to Stackoverflow.
- 3) We present a solution based on analyzing textual features (textual similarity between bug report title and Stackoverflow questions) and contextual features (such as question tags representing the topic) to recommend a Stackoverflow question in response to a bug report. We believe that a recommendation engine (tool support) that automatically suggests relevant Stackoverflow knowledge-base to developers can save time during bug resolution. We present the proposed solution and empirical results ((performance evaluation and validation)) demonstrating the effectiveness of the method on dataset containing the ground-truth.
- 4) A survey conducted with experienced Software Maintenance Professionals on the topic of integrating issue tracking system with community driven Q&A websites.

We believe that there is a dearth of academic studies surveying the needs, problems encountered, human-factors and suggestions on the problem area discussed in this paper.

There are many bug tracking systems available in the industry to use. Bug tracking systems are also called as issue tracking system or issue reporting system or defect tracking system or defect reporting system, etc. Bug tracking systems are developed by open source community as well as closed source organizations as a proprietary software. Open source means the source code is being shared with everybody under the General Public License (GPL) policy. Anyone can contribute to the code voluntarily. There is no restriction towards the submission of code. The moderator will see and verify the reputation of the code submitter through his code submission pattern and its status can also be changed as moderator. While in closed source community, the source code is the property of the organization and the people who are outside the project may not be able to see/browse the code. Outsiders can submit only bugs through the feedback or e-mail to the sales/support person specified by the organization. In this article, we will consider the open source project development scenario except one or two closed source products. Some of bug tracking tools are from open source communities and some of them from closed source communities or commercial organizations. There are organizations which can also provide support for the open source solutions.

III PROPOSED SYSTEM

The Proposed System we have the end-user dataset excel sheet, and we will enter the parameter randomly, it will predict the result on the basis of status of the bug. Benefit of the system we can check and remove the max probability parameter. Although bug reporters may not recognize that the bug they are describing is a security bug, the natural-language description of the bug in the BR may be adequate to indicate that the bug is security-related and thus the BR is an SBR.

- a. Feature selection is iteratively performed until optimum points are reached. At the end of Step 5, there is a reduced feature set that performs optimally for the chosen classifier metric.
- b. Using the reduced feature set, a classification model is trained. Although many classification techniques could be employed, this paper focuses on the use of Naive Bayes.

Its Advantages over Existing System are The performance is increased due to well designed database, Security is increased, Time saving in report generation and Easy to update the details.

There are three main steps. The first step is to obtain a labeled BR data set that contains textual descriptions of bugs and labels to indicate whether a BR is an SBR or an NSBR. The labeled BR data set is required for building and evaluating our natural-language predictive model. The second step is to create three configuration files that 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 estimates the probability that a BR is an SBR.

IV DATA FLOW DIAGRAMS

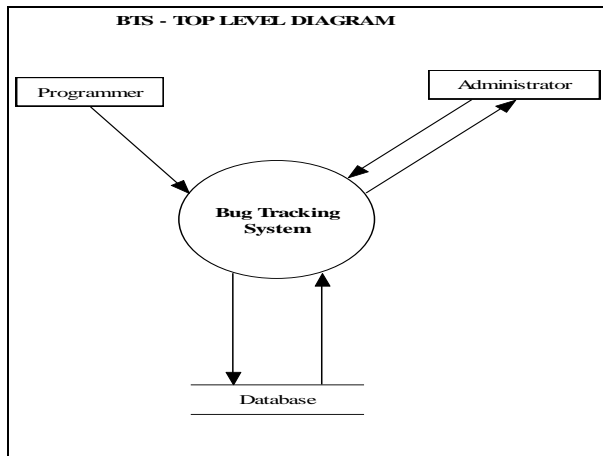


Fig 1 : BTS Top Level Diagram(a)

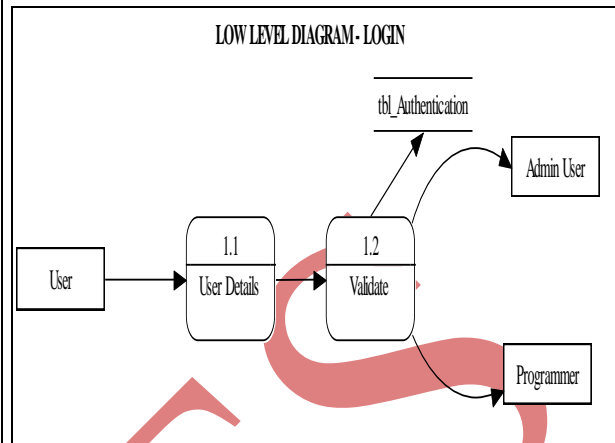


Fig 2 : Low Level Diagram-Login

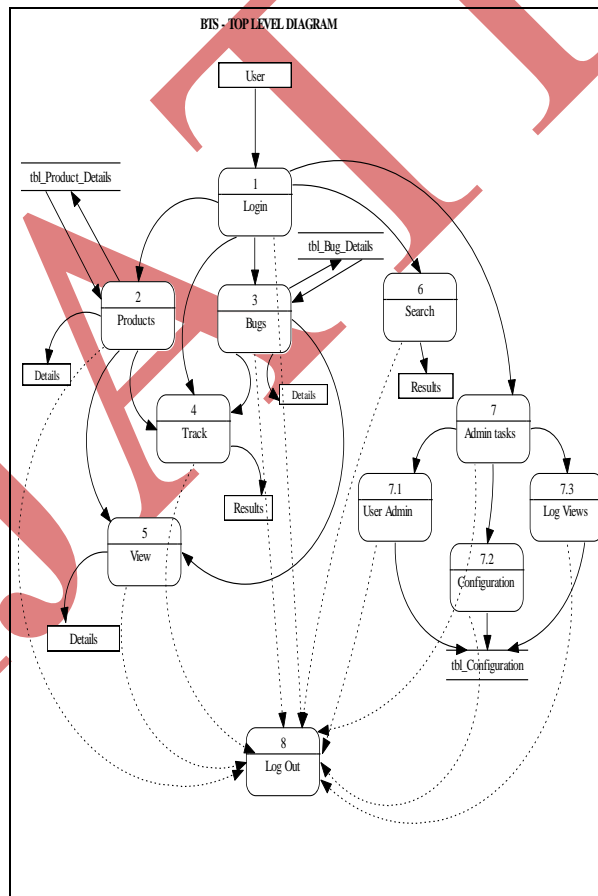


Fig 3 : BTS Top Level Diagram(b)

V RESEARCH METHODOLOGY

First find the likelihood of the two classes

- For "yes" = $2/9 * 3/9 * 3/9 * 3/9 * 9/14 = 0.0053$
- For "no" = $3/5 * 1/5 * 4/5 * 3/5 * 5/14 = 0.0206$
- Conversion into a probability by normalization:
 - $P(\text{"yes"}) = 0.0053 / (0.0053 + 0.0206) = 0.205$
 - $P(\text{"no"}) = 0.0206 / (0.0053 + 0.0206) = 0.795$

5.1 Bayes' Rule

More generally, the above is just an application of Bayes' Theorem.

- Probability of event H given evidence E:

$$\Pr(H | E) = \frac{\Pr(E | H) * \Pr(H)}{\Pr(E)}$$

- A priori probability of H= Pr(H)
 - Probability of event before evidence has been seen
- A posteriori probability of H= Pr[H|E]
 - Probability of event after evidence has been seen
- Classification learning: what's the probability of the class given an instance?
 - Evidence E = instance
 - Event H = class value for instance

- Naive Bayes assumption: evidence can be split into independent parts (i.e. attributes of instance!)

$$\Pr(H | E) = \frac{\Pr(E1 | H) * \Pr(E2 | H) * \dots * \Pr(En | H) * \Pr(H)}{\Pr(E)}$$

- We used this above. Here's our evidence:

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

- Here's the probability for "yes":

$$\begin{aligned} \Pr(\text{yes} | E) &= \Pr(\text{Outlook} = \text{Sunny} | \text{yes}) * \\ &\quad \Pr(\text{Temperature} = \text{Cool} | \text{yes}) * \\ &\quad \Pr(\text{Humidity} = \text{High} | \text{yes}) * \Pr(\text{yes}) \quad \Pr(\text{Windy} = \text{True} | \text{yes}) * \Pr(\text{yes}) / \Pr(E) = (2/9 * 3/9 * \\ &\quad 3/9 * 3/9) * 9/14 / \Pr(E) \end{aligned}$$

Return the classification with highest probability

- Probability of the evidence $\Pr(E)$
 - Constant across all possible classifications;
 - So, when comparing N classifications, it cancels out

5.2 Missing Values

Missing values are a problem for any learner. Naive Bayes' treatment of missing values is particularly elegant.

- During training: instance is not included in frequency count for attribute value-class combination
- During classification: attribute will be omitted from calculation

Eg.: Outlook Temp. Humidity Windy Play
? Cool High True ?%%

- Likelihood of "yes" = $3/9 * 3/9 * 3/9 * 9/14 = 0.0238$
- Likelihood of "no" = $1/5 * 4/5 * 3/5 * 5/14 = 0.0343$
- $P(\text{"yes"}) = 0.0238 / (0.0238 + 0.0343) = 41\%$
- $P(\text{"no"}) = 0.0343 / (0.0238 + 0.0343) = 59\%$

To fulfill our purposes, we propose our own procedures that may be useful for various domain areas.

Procedure 1: PPR. Mining Specific Client's Bug Usage

Input: Database D of transactions, Specific Product attributes

Output: Sites (info) used by individual user

Method

1. Accept input_Atr (Specific Attribute type)
2. for (int i=0; i<= D.size; i++)
3. if (input_Atr == D_Atr)
4. extract information as info from database
5. Return info

6.end

For the Web Site Maintainers, they should know the users' interest rate on their sites. Depending on the users' interest rate decreasing or increasing through the time, they can modify their site structure to attract more users from the aspect of economic benefits. We proposed the procedure AM especially for maintainers and developers as a result of Web usage over a specific period of time.

Procedure 2: Find the users' Component attribute to evaluate all state in dataset

Input: Database D of transactions, Specific component

Output: total transaction of Component attribute

Method

```
3. count= 0; temp[ ] =null; // initialization
4. while (component_state= =NSBR || component_state==SBR)
5. if( temp[ ] = =null)
6. temp[ ] = D_Atr
7. count ++
8. else
9. while (temp [ ])
10. if (temp [ ]== D_Atr )
11. do nothing
12. else temp[ ] = = D_Atr
13. count ++
14. return count // total number of component attribute for class dependency
```

VI SIMULATION RESULTS

If the model classifies an SBR as an NSBR, or if the model classifies an NSBR as an SBR, then the result is a misclassification. A true positive (TP) is a verified SBR that is correctly classified by the model. A false positive (FP) is a verified NSBR that is incorrectly classified to be an SBR. A false negative (FN) is a verified SBR that is incorrectly classified to be an NSBR. A true negative (TN) is a verified NSBR that is correctly classified to be an NSBR. Model precision is the percentage of correctly classified SBRs among SBRs and NSBRs that have been classified by the model to be SBRs (i.e., exceeding a minimum probability).

We did not reveal the estimated probabilities to the security engineers to reduce potential bias in their analyses. Based on prior discussions with the security engineers, we estimated that security engineers would require

approximately 175 person-hours to analyze Apilot and determine whether the manually-labeled NSBRs are actually SBRs. If two security engineers disagreed on their evaluations of a manually-labeled BR, then they discussed their differences and reached an agreeable consensus.

Now we get the finite stage for the dataset , which depend on the product parameter, component parameter, status parameter, and resolution parameter. Corresponding to the SBR and NSBR class, we getting the fix stage to whole dataset.

ID	Product	Component	Status	Resolution	Summary	Category
27194	Core	Security FSM	VERIFIED	FIXED	When going from a secure to a non-secure page without clicking a button in the security dialog, the non-secure...	NSBR
51978	Core	Security UI	DUPLICATE		loading non-secure page in the untrusted view causes a subsequent secure page to appear non-secure.	NSBR
59447	Firefox	Security	RESOLVED	DUPLICATE	Secure site that appears as secure but does secure connection	NSBR
30254	Core	Networking HTTP	RESOLVED	DUPLICATE	External CSS file is not rendered for secure page when entering secure page from non-secure page. #4021...	NSBR
45032	Thunderbird	Account Manager	RESOLVED	FIXED	Intermittent failure for secure authentication failure on secure connections during account verification	SBR
64275	Core	Security UI	NEW	DUPLICATE	show modal content indication on secure pages that use cookies without the secure flag	NSBR
85431	Core	Plugins	VERIFIED	FIXED	New Firefox when opening a pdf file from a secure non-secure link	NSBR
12321	Core	Security UI	VERIFIED	INVALID	Closing a window containing a secure page should not warn about leaving a secure site	NSBR
13916	Core	Security	RESOLVED	WORKAROUND	Secure lock icon doesn't display "locked" when in secure site	SBR
42832	Firefox	Security	RESOLVED	INCOMPLETE	To ensure from a secure page to one that is not in the list, all secure the color that is secure	SBR
48347	Firefox	General	RESOLVED	INCOMPLETE	Reason when pop-up window "This page contains secure and non-secure items..." appears	SBR
24312	Core	Security UI	VERIFIED	WONTFIX	Add secure site notification to pages that point to secure address	NSBR

Fig 4 : Bug Tracking Loaded Dataset

Fig 5 : Attribute parameter value for dataset

After classification it will getting the total category corresponding to the dataset in front of such attribute , the total category firefox is 15 in front of SBR and 224 for NSBR , in Component block security is 5 times occurrence in front of SBR and 134 times occurrence NSBR.

The maximum probability for NSBR is 0.001951 , with respect to firefox, security, berified, fixed respective to whole dataset

Product: Firefox	SBR: 15 NSBR: 224	Probability For SBR: 0.001129035173212
Component: SECURITY	SBR: 5 NSBR: 134	
Status: VERIFIED	SBR: 22 NSBR: 240	
Resolution: FIXED	SBR: 28 NSBR: 443	Probability For NSBR: 0.0195176942911743
Count Category	Calculate SBR: SBR	Calculate NSBR: NSBR
		Total Category: 1193

Fig 6 : Find Fix State Value For The Dataset

Fig 7 : Fix Probability For NSBR Value

After changing the attribute we gain the different probability ,now we have to consider the maximum probability to whole dataset.

After gaining the words probability, we fetch the summary for text mining in bug tracking system . Now we have to evaluate the bug related text and remove the stop words.

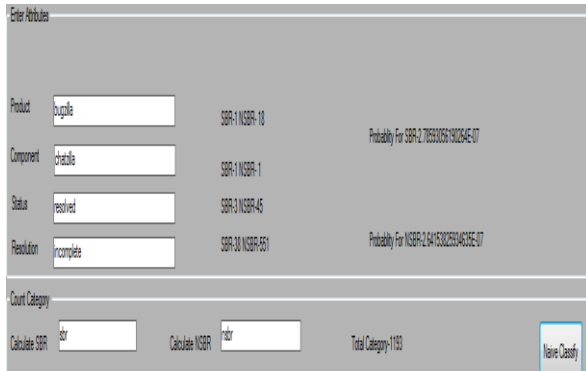


Fig 8 : Predictive iteration for SBR and NSBR

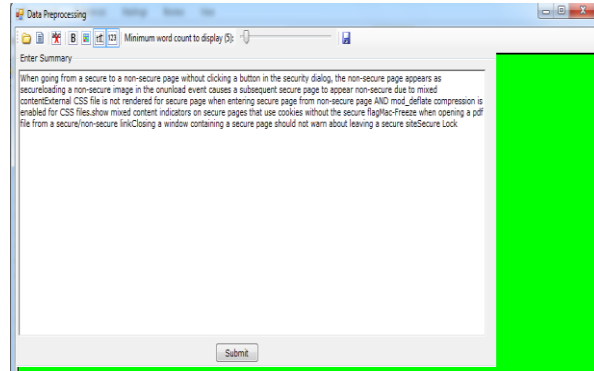


Fig 9: Text data processing for stop word count

Get the maximum probability for words occurrence . preprocessing the data get count the words. After preprocessing we start for the stemming of words which we have to consider only .

Evaluation for the bug fix prediction which efficiency with respect to time improving the efficiency average rate scale is 220-150 .which is moreover efficient for the other predictive system.

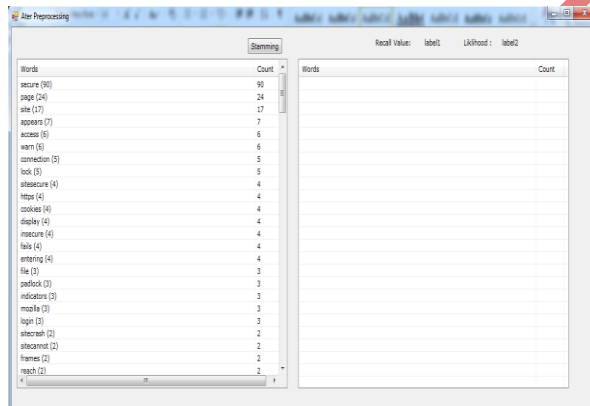
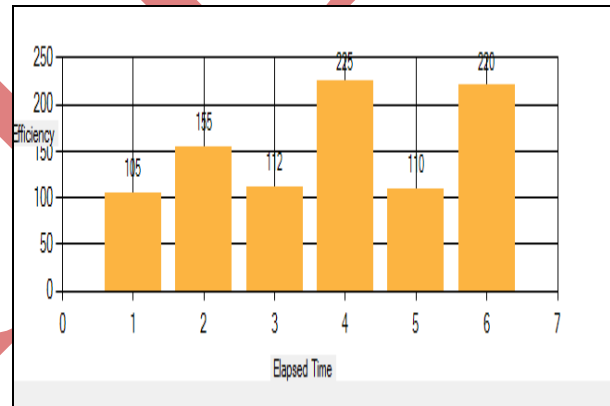
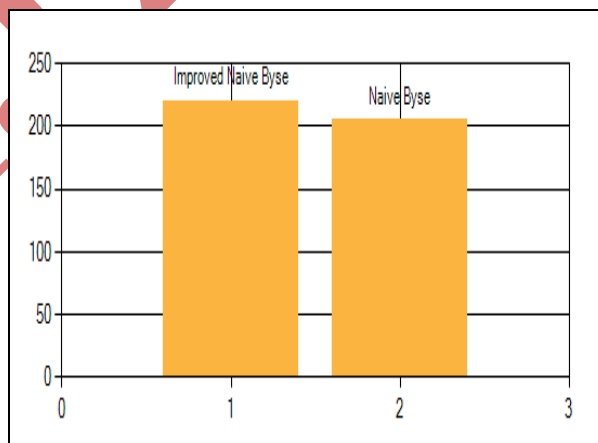


Fig 10 : Stop words list



Graph 1 : Performance for improved naive with respect to efficiency of BTS



Graph 2 : Comparison between INYB and NYB with respect to Efficiency

VII CONCLUSION AND FUTURE SCOPE

Current bug tracking systems do not effectively elicit all of the information needed by developers. Without this information developers cannot resolve bugs in a timely fashion and so we believe that improvement to the way bug tracking systems collect information are needed.

While implementing a range of improvements may be ideal, bug tracking systems may instead prefer to specialize, thus providing a rich set of choices. This would be a healthy change to the current situation where they all provide identical functionality. Identify information needs in a large sample of bug reports through manual inspection. This will help to compile a catalog of questions that can be used for the expert system. Using this catalog, collect answers and defect locations for another large sample of bug reports. This dataset will be used to automatically learn a prediction model. Evaluate the predictions and conduct usability studies.

REFERENCES

- [1] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An Approach to Detecting Duplicate Bug Reports Using Natural Language and Execution Information" Proc of the ICSE, pp. 461-470, 2008.
- [2] I. Witten and E. Frank, Data Mining, Second ed. San Francisco, Elsevier, 2005.
- [3] Guzella, T. S., Mota-Santos, T. A., Uchoa, J. Q., & Caminhas, W. M. (2008). Identification of spam messages using an approach inspired on the immune system. Biosystems, 92(3), 215–225.
- [4] Gyongyi, Z., & Garcia-Molina, H. (2005). Web spam taxonomy. In Proceedings of the first international workshop on adversarial information retrieval on the web (AIRWeb).
- [5] Haider, P., Brefeld, U., & Scheffer, T. (2007). Supervised clustering of streaming data for email batch detection. In Proc of the int conf on mach learn.
- [6] Hastie, T., Tibshirani, R., & Friedman, J. (2001). The elements of statistical learning. Springer. Hayes, B. (2007). How many ways can you spell V1@gra? Scientific American, 95(4), 298–302.
- [7] Haykin, S. (1998). Neural networks: A comprehensive foundation (2nd ed.). Prentice Hall.
- [8] He, J., & Thiesson, B. (2007). Asymmetric gradient boosting with application to spam filtering. In Proc of the fourth conf on email and anti-spam.
- [9] Hoanca, B. (2006). How good are our weapons in the spam wars? IEEE Technology and Society Magazine, 25(1), 22–30.
- [10] Hsiao, W.-F., & Chang, T.-M. (2008). An incremental cluster-based approach to spam filtering. Expert Systems with Applications, 34(3), 1599–1608.

- [11] Jones, R. (2003). Spam. <<http://www.annexia.org/spam>> (visited on July 2008). Jorgensen, Z., Zhou, Y., & Inge, M. (2008). A multiple instance learning strategy for combating good word attacks on spam filters. *Journal of Machine Learning Research*, 8, 993–1019.
- [12] T. Kalt. A new probabilistic method of text classification and retrieval. Technical Report IR-78, Center for Intelligent Information Retrieval, University of Massachusetts, Amherst, MA, 1996.
- [13] A. McCallum. Multi-label text classification with a mixture model trained by EM. In *AAAI Workshop on Text Learning*, 1999.
- [14] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering, 1996.
- [15] K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proc. of AAAI 1998*, pp 792– 799. AAAI Press.

UNACCEPTED