Duplication Detection for Software Bug Reports based on Topic Model

Jie Zou[b], Ling Xu[a,b,*], Mengning Yang[a,b], Meng Yan[b], Dan Yang[b], Xiaohong Zhang[a,b]
a. Key Laboratory of Dependable Service Computing in Cyber Physical Society Ministry of Education, Chongqing 400044, PR China
b. The School of Software Engineering, Chongqing University, Chongqing (401331), PR China.
xuling@cqu.edu.cn

***Abstract*—The traditional duplicate bug reports detection approaches are usually based on vector space model. However, the experimental result is rarely satisfying since this method cannot distinguish semantic correlation among bug reports which written by natural languages. Topic model, as a method to model underlying topics of texts, can solve the problem of document similarity calculation methods used in the information retrieving. It can find the semantic topics among the texts through massive training data, and obtain semantic relatedness among documents. Therefore, this paper proposes a novel duplication detection method based on topic model. Through selecting bug reports with execution information and combing with classified information of bugs, not only does this new method overcome the problem of high dimension, sparse data and loud noise, but also avoid the problem of synonymy and ambiguity in the natural languages. Comparing to the traditional SVM method, the recall rate and precision rate of our proposed approach have obviously increased, which indicates the effectiveness of this new method.**

Keywords—duplicate bug reports detection, topic model, execution information, vector space model

I. INTRODUCTION

Along with the increase of the size of software project, software becomes more and more complicated. Maintenance expense takes up 2/3 of software life cycle expense. Software bug reports are document descriptions about possible defects or errors made by software testers or users while maintaining. Along with the increasing scale and updating versions, various users submit bug reports to bug-tracking management system every day. Thus, open source software like Eclipse, Firefox, Open Office and so on, produce a great deal of duplicate bug reports. Take Firefox as an example, the percentage of duplicate bug report in its software bug database is up to 30% [1]. "Everyday, almost 300 bugs appear that need triaging. This is far too much for only the Mozilla programmers to handle", a programmer from Mozilla reported in 2005. Duplicate bug reports not only occupy space, but also cause problems with bug report assignment, increasing labor costs.

In order to relieve the burden of manual detection on the duplicate bug reports, more and more researchers involve themselves into this field currently. Runeson P [2] and his co-workers, took bug reporter library of Sony Ericsson Mobile Communications as a dataset, calculate similarities of bug reports after vectoring and normalizing processing, and gained 30% precision. Wang X Y [3] et al. added execution information on the basis of Runeson P's research, defined two kinds of similarities among bug reports: natural language similarity and execution information similarity. If take executive information as the only reference standard, the recall rate should be around 93%, and the precision rate 67%. Based on the research of Runeson P et al, Sun [4] et al. mapped the vectorized and identified bug reports to discriminative model, and then trained a SVM classifier to detect duplicate bug reports. Compared with Runeson P's method, Sun's precision rate is about 20% higher, but it's lower than Wang X Y's. Thus it can be seen that introduction of execution information can enormously raise the precision rate of detection. Standardized execution information made automatic detection of duplicate bugs available not only in theoretical research but also in practice. Compared with the natural language, standard executive information can be more reliable and understandable to describe the situation when bugs emerge. Moreover, to facilitate standardization, procedurization, and sophistication is the inevitable direction of software engineering development, thus, this paper chooses executive information bug reports as the main research object.

Most of the detecting methods of similarities are based on one hypothesis: the more words are repeated in different documents, the more similar these documents are. However, it is not exactly so in practice. Very often, correlation degree depends on semantic relation, instead of repeating words and expressions. Topic model LDA [5] (Latent Dirichlet Allocation) can find these semantic relations of bug reports. LDA, is a generated topic model which provides an available method for auto-organization, comprehension, and massive documents detection. LDA can be used to explore the hidden topics in corpus, mark on the document according to the topics, and finally organize, classify, induce and retrieve similarity among documents on the basis on these marks. Also, during the process of submitting bug reports, developers or users need to specify the category of these bug reports which refers to classified information defined by submitters. The classified information reflects the basic properties of bug reports on a certain degree. So we put weight on it and combine it with descriptive information to

2165-3836/16 $31.00 © 2016 IEEE
DOI 10.1109/ICSS.2016.16

improve the duplicates detection results. In this paper, a new LDA-based method is proposed to detect similarities of documents, classified information submitted by the original users, executive information, and descriptive information of bugs are combined together to obtain the final experimental result. The main idea of this paper is as follows: firstly, build a data warehouse which is taken as training data, and use LDA method to construct a topic model; then, construct a small testing sample space, and make a thematic inference according to the constructed topic model to obtain the document-topic matrix, thus map the bug report documents from traditional high dimension word space to low dimensional topic space; at last, calculate similarities among documents in low dimension topic space. If the similarity is greater than the set threshold, the bug reports will be considered as duplicates.

The remainder of this paper is organized as follows:

Chapter 2 explains our approach for duplicate bug report detection in detail. Chapter 3 shows and evaluates the experimental results. Chapter 4 analyzes the threats to validity. Chapter 5 discusses the related work. Chapter 5 draws a conclusion of the research work.

II. THE PROPOSED METHOD

Our approach is consisted of three key steps. Firstly, select bug reports from the experimental data set as the training sample space, and then build a topic model by using LDA method in the training sample space; secondly, randomly select some bug reports as the test sample space data, extract classified information of each bug in the sample space, and meanwhile deduce the topics of the testing samples by the constructed topic model to get a corresponding topic-document matrix; finally, transform the document-topic matrix into the vector representation of document object in the topic space, calculate the similarity between documents by calculating the cosine of the angle of vector space and put weight on it, and at the same time, calculate the similarity of classified information and put weight on it. The multiplication of the two is the final results of the bug reports similarities, and then set a threshold, the bug reports will be considered as duplicates when the similarity is greater than the threshold. The overall procedure of our method is shown in figure 1.

A. Constructing Topic Model

The method of duplicate bug reports detection presented in this paper is based on LDA, which models the bug reports as documents. LDA model is a complete generative model which regards blend weights of topic as random variables of T dimension parameters rather than set of individual parameters directly related to the training data. It avoids the shortcomings of PLSA model [6].

LDA model assumes a process of producing a document firstly, and then observes and predicts the production process. LDA supposes that all documents have K topics (the topic is represented by the word distribution in LDA model). To generate a document, the first step is to generate a topic distribution of the document, and then assemblage of words; to generate a word, a topic should be randomly chosen according to the conditional distribution of topics w.r.t the document, and after that a word is randomly chosen from the conditional distribution of words w.r.t the topic.

Assuming that the K dimensional vector α is the parameter of the prior data distribution of the topic, $K \cdot V$ matrix β is the parameter of the distribution of the words (V is the sum of words) in the topic, i.e. $\beta_{ij}=p(w_j|z_j)=$ the probability of the word w_j in the i_{th} topic. Thus, we generate a document topic distribution and N topics, and the probability of the N words we get in this document is expressed as:

$$p(\theta,z,w)=p(\theta|\alpha)\prod_{n=1}^{N}p(z_n|\theta)\,(w_n\,|z_n,\beta) \tag{1}$$

Where θ is the topic distribution vector of the document, z is the topic vector of N dimensions, and w is the vector composed of the N words. Since θ and z are latent variables in the training data which cannot be observed, they are eliminated through marginal distribution from the left:

$$p(w\,|\,\alpha,\beta)=\int p(\theta|\alpha)\prod_{n=1}^{N}p(z_n|\theta)\,(w_n\,|\,z_n,\beta)\mathrm{d}\theta \tag{2}$$

To corpus D which has M documents,

$$p(D|\,\alpha,\beta)=\sum_{d=1\cdots M}p(w_d\,|\alpha,\beta) \tag{3}$$

Thus

$$p(D|\,\alpha,\beta)=\prod_{d=1}^{M}\int p(\theta_d|\alpha)(\prod_{n=1}^{N_d}\sum_{z_{dn}}p(z_{dn}|\theta_d)\mathrm{p}\,(w_{dn}\,|z_{dn},\beta))\mathrm{d}\theta_d \tag{4}$$

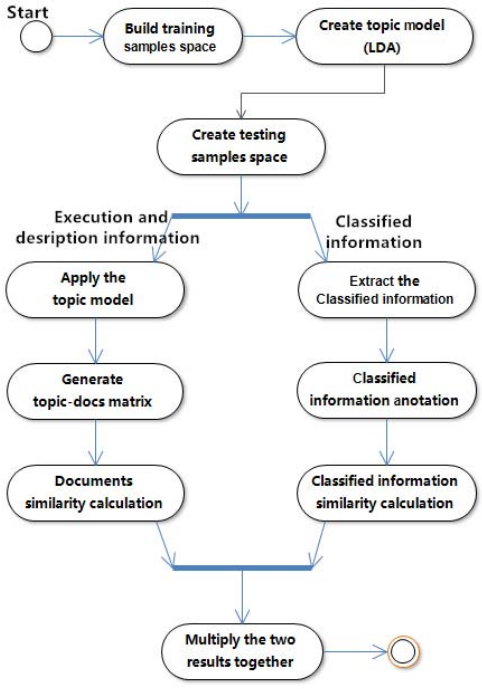

Fig 1. The overall procedure of the method

As shown above, the process of constructing the LDA model is a process of achieving maximized parameters α and β of $p(D|\alpha,\beta)$.

B. Applying LDA Model and Extracting classified Information

We obtain the conditional distribution of words within the topics $p(w|z)$ and the conditional distribution of topics within documents $p(z|d)$ of training samples after constructing a topic model. When there is a new submitted bug report in testing samples, we deduce the topics of the new unlabeled texts by the trained topic model and get its corresponding topic distribution $p(z|d_{new})$ through the previous $p(w|z)$. And then we can calculate the similarities between the new bug report and the bug reports of training samples by formula (6) after getting the $p(z|d)$ and $p(z|d_{new})$. The corresponding topic distribution $p(z|d_{new})$ is also the conditional probability distribution of the document in the topic space.

$$p(\theta,z|w,\alpha,\beta)=\frac{p(\theta,z,w|\alpha,\beta)}{p(w|\alpha,\beta)} \tag{5}$$

After application of sample space model, extract classified information from each bug report in sample space and mark them for the follow-up experiment. The classified information of bug report contains four kinds of information: classification, product, component, and version. All kinds of information are in relation to progressive gradual progressive relation.

C. Similarity Calculation

Similarity calculation is divided into two steps: the first is the similarity of each document in topic space, on which is weight 1 placed; the second is the similarity calculation of classified information of each document, on which is weight 1 placed, too.

After application of topic model to the testing sample, we can get its conditional probability distribution in the topic space in the form of a document-topic matrix which can easily be transformed into vector representation of documents in the topic space. Thus, we can express semantic similarity by calculating the similarity between the documents in topic space.

There are a lot of methods to measure vector space similarity. The one this paper adopts is to calculate the angle cosine value of vectors. To calculate the similarity of testing sample document *D1* and *D2*, the vector of document *D1* in the i_{th} dimension topic space is d_{1i}, and that of document *D2* is d_{2i}. The similarity calculation formula is as follows:

$$Sim(d_1,d_2)=\frac{\sum_i (d_{1i}*d_{2i})}{\sqrt{\sum_i d_{1i}^2 * d_{2i}^2}} \tag{6}$$

Note that, the cosine value of the vector angles ranges between 0 to 1, and the bigger the angle is, the smaller the cosine value is. From the space relationship corresponding to the similarity, we can find that when the cosine value is bigger and the angle is smaller, space similarity or semantic similarity between the two documents is higher.

Classified information of document, which provides the basis of similarity for detecting duplicate bug report, is a basic feature of bug report. Classified information is a kind of structured information organized according to the classification\ product\ component\ version sequence, so the appropriate way to calculate similarity of classified information is not to compare the classified information according to unstructured text data but layer sequence, and stop comparing when find the difference at the first time to stop. The similarity of descriptive information is between 0 and 1. Set the weight of classified information at 1 and the number of the same level obtained through comparison are 0.4, 0.55, 0.7, 0.85, and 1.

III. Experimental Results and Analysis

In this experiment, two problems should be taken into consideration. First, under what condition can achieve optimum experimental results? Second, how well the results compared with experiments in other methods?

To evaluate the results, we used the recall and precision rates as standard metrics.

$$RecallRate=\frac{N_{detected}}{N_{total}} \tag{7}$$

$$PrecisionRate=\frac{N_{detected}}{N_{detectedall}} \tag{8}$$

Where $N_{detected}$ is the number of correct experimental detections in the duplicate bug reports of the test sample, N_{total} is the total number of the actual duplicate bug reports in the testing sample, $N_{detectedall}$ is the total number of duplicate bug reports in the experimental detections (including the right and wrong).

A. Dataset and data Preprocessing

When selecting the experimental data set, we take two factors into consideration:

1. There is often a bug correction period after main versions release of a software (usually one and a half months).

2. The most effective time to detect duplicate bug reports is within 50 days when the corresponding bug source reports are submitted.

According to the above principles, this experiment is based on the bug report data from the famous open source project Eclipse, taking bug reports newly submitted in 3 months from June 26, 2006 to September 26, 2006 as the training sample space (due to the updating of version 3.2 Callisto issued by Eclipse on June 26, 2006). Testing samples in the later part of the experiment are from this training sample space.

Before constructing topic model for the training sample space, we need to format the original data. Since the original report file is organized in the form of XML, and also

contains a lot of redundant information, only bug title and content are selected as the main experimental data. If the title and content of the original bugs are all null, which infers that this report is non-existent, it should be removed, such as bug report ID 511.

For the training sample space, there were 10400 original bug reports. After removal of invalid bug reports, 9600 remained, among which 1115 were marked “duplicate”, which accounts for 11.6% of the total. Whether it is marked as “duplicate” is decided by the value of "resolution" in the original XML file. And classified information is obtained through classification\ product\ component\version and marked to each bug report.

The preprocessing is followed by the treatment of extracted data including data cleaning, segmentation, stem extraction and removal of stop words. Note that, there are a data cleaning process differ from common text processing, because most of the description of the bug has invalid information expressed in fixed sentence pattern, such as:

1 : Fixed in HEAD.

Available in builds > N20071108-0010.Available in builds > N20071108-0010.

2 : *** Bug 208441 has been marked as a duplicate of this bug. ***

The sentence pattern 1 shows that the bug has been revised in previous versions, while patterns 2 point out which bugs are marked as duplicate bugs. The two patterns mentioned above has no connection with the bug itself, since there is a lot of this kind of information in the original file and it produces a large number of duplicate invalid information which will eventually have a great influence on the experimental results, they are removed according to the fixed patterns firstly. After finishing the cleaning of the data, the few remaining preprocess steps can be carried out to further refine information.

A standard data set will be obtained when data preprocessing is complete. Then, we construct a LDA topic model in the training sample space.

To construct the LDA model, we used natural language processing toolkit MALLET [7] open source (Machine Learning for language Toolkit) to implement LDA modeling. First, we convert all texts of training sample space to feature sequence, then set the number of topics according to the size of sample space, and finally, get a topic model of training samples.

To construct a test sample space, we randomly select 22 bug reports with executive information from 1115 which have been marked duplicate in the training sample space, and choose 178 from non-duplicate reports to construct a basic database. Get actual duplicate bug reports corresponding to the 22 bug reports from the website of Eclipse, add them to the basic database. In the experiment of this paper, 47 are added externally. Finally, we get a small testing sample space containing 247 bug reports. Apply the constructed topic model in this test space, and get the document-topic matrix.

All the document-topic matrixes are converted into vector space, and calculate the similarity between the bug reports in the testing sample space by formula (6). And then we set a threshold, the bug reports will be considered duplicates when the similarity is greater than the threshold value.

B. Experimental results and analysis

As for the bug reports with executive information, Wang X Y [8] et al defined two kinds of similarities between them: natural language similarity and executive similarity, estimated the similarity of bug reports through the distance between vectors after vectoring(TF*IDF) and normalizing. Wang X Y et al. achieved a 95% recall rate and the highest precision rate 67% using this method. What have been found from this experiment is that increase of the recall rate is at the expense of precision rate [8]. Instead, this paper applies a topic model, calculates the corresponding semantics similarities between bug reports through the topics extracted by LDA model. Also, we calculate the similarities of classified information at the same time, place weight on semantic similarities and classification similarity and finally get the recommendation results.

Figure 2 shows the results of our method on testing samples in comparison with Wang X Y’s. And the threshold is set as 0.95 in figure 2. Since the recall rate parallels Wang X Y’s, but the precision is greatly improved at the same time.

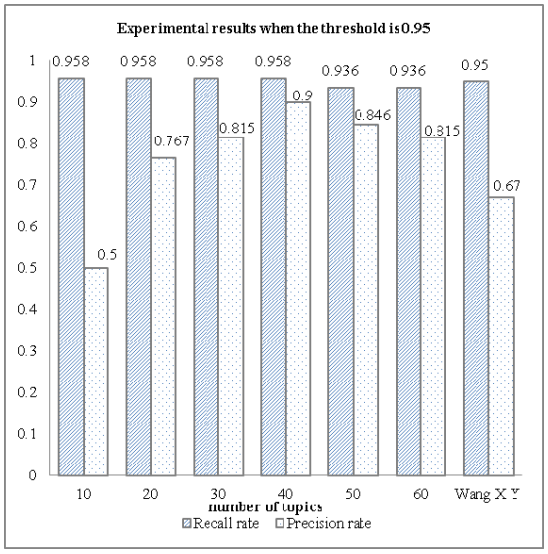

Fig 2. The comparison of our results with Wang X Y

As can be seen from figure 2, compared with the traditional SVM method adopted to detect duplicate bug report with executive information, the recall rate is roughly the same at about 95%, but the precision rate is improved greatly. As for this experiment, when 40 topics are selected, the precision rate is up to 90%, obviously increased compared with 67% by traditional method.

From figure 2 we see that the parameter selection will directly affect the quality of the modeling and the experiments results when building models for specific sample spaces. We experiment the influences of different parameters on the experiment results, such as the number of

topics and threshold. Figure 3 shows the influence of topics amount on the experiment results, As shown in Figure 3, when the number of topics reaches 40, the precision rate is the best, and the recall rate at a high level.

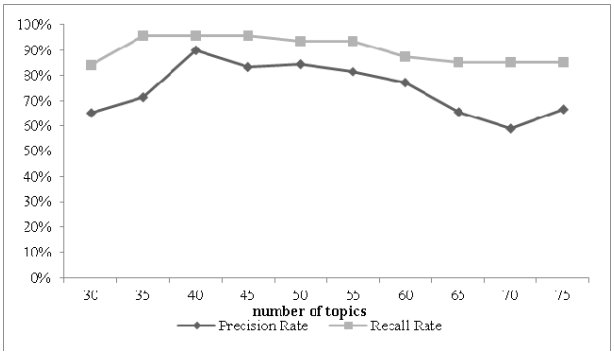

Fig 3. The influence of topics amount on the experiment results

Therefore, when there are too few topics, the discrimination between topics is not obvious enough and the particle size distribution is insufficient. Too many topics usually make it dispersive, which may lead to unsatisfying results. Only when the amount of topics set to a reasonable interval, the experimental results can be optimal.

Similarly, the threshold is also important for the performance of the model. Figure 4 and Figure 5 shows the recall and precision rate when the threshold is set to different value.

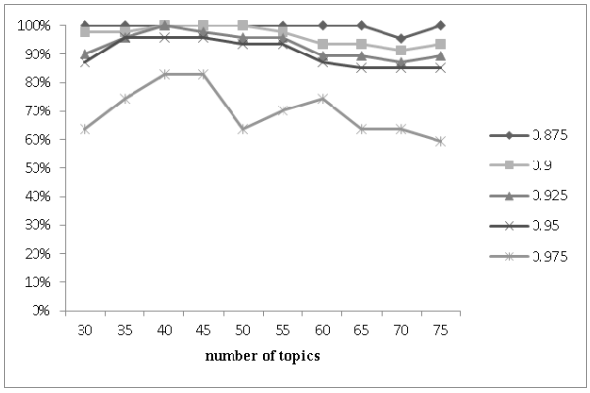

Fig 4. The comparison with recall rate at different threshold

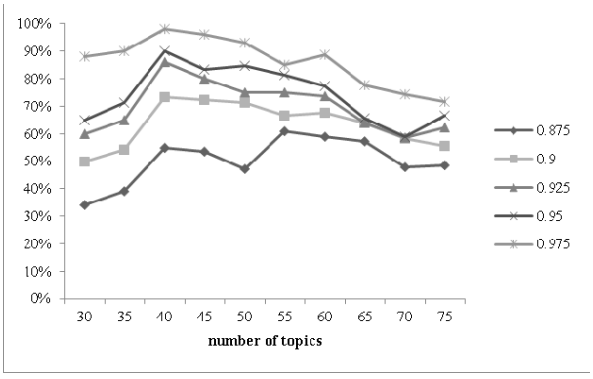

Fig 5. The comparison with precision rate at different threshold

As shown in Figure 4, the smaller the threshold is, the more recommendation results which are greater than the threshold is. Thus, it leads to a greater probability of accurate results and a higher the recall rate.

In Figure 5, when the threshold is higher, the results obtained in the experiment set are small, which results from the less invalid results, the precision is higher.

In a comprehensive analysis of Figure 4 and Figure 5, the smaller the threshold is, the higher the recall rate becomes, but the lower precision rate is; the greater the threshold is, the lower the recall rate becomes, but the precision rate is increased. According to the experiment, when the threshold is at 0.95, the increase of precision rate is greater than the decline of recall rate. The optimal experimental results come out when the topic number is 40, the threshold is 0.95.

As mentioned above, the different values of parameters affect the detection results. There is no provably optimal choice for the parameters [9] [10]. The choice is a tradeoff between coarser parameters (smaller value) and finer grained parameters (larger value). So a reasonable choice should be made which is mainly dependent on the experiments and experience of the experimenter.

IV. THREATS TO VALIDITY

In our study, we assume that the information provided by the bug reporter is correct, including the textual, categorial, and duplication information. If a bug report information is not enough or misleading, the performance of our approach is adversely affected.

When performing the LDA model, we chose to use 40 as the number of topics. There is no absolutely optimal choice for the number of topics in all situations and all datasets [9] [10]. The choice is to seek a reasonable value between coarser-grained and finer-grained. To alleviate this threat, we experimented with different values and chose the one that gave the best results by considering the recall rate and precision rate.

The next validity threat is limited by the sole use of the Eclipse open source bug repository, but this is very large and popular with tens of thousands of developers worldwide. and since Eclipse uses Bugzilla as its bug tracking system, which is the most widely used bug tracking system, the breadth of the Eclipse sub-projects provides some form of generality. In the future, we would consider more software systems, especially on commercial projects.

We tried to minimize the internal threats to validity by using mature tools for extracting data, executing LDA, duplication detection and visualization. We used the SAX API to parse and extract data, MALLET to construct our LDA model, JAVA programming environment for duplication detection, Excel and Matlab to visualize figures.

V. RELATED WORK

To detect the duplicate bug reports, an increasing number of experts currently involve themselves into this field. Most of the early works were statistical IR approaches. Hiew et al. [11] applied VSM to detect duplicate bug reports, which models a bug report as a vector. Sureka and Jalote [12] used a character N-gram-based model to detect duplicate bug

reports, which applied N-gram at the character level. Sun et al. [13] proposed an REP model, an advanced IR approach, to calculate the similarity between two bug reports. They also extended BM25F [14], a document similarity formula built upon Tf-Idf.

Runeson P et al. [2] applied Natural Language Processing (NLP) on the bug reports. Wang X Y et al. [3] added execution information on the basis of Runeson P.'s research.

The Machine Learning (ML) method is also a popular method on duplicate bug reports detection. Jalbert and Weimer [15] proposed a system that automatically classifies duplicate bug reports to save developers' time. They used textual semantics, surface features, and graph clustering to predict duplicate status. Tian et al. [16] extended Jalbert and Weimer's work [15] by utilizing REP. And Sun et al. [4] utilizes a discriminative model for duplicates detection.

VI. CONCLUSIONS

A new detection of duplicate report based on topic model is proposed in this paper. Through topic model, problems can be solved by transforming them from the original words space to topic space. We also consider the executive information and the classification of bug reports in our proposed method.

The experimental results show that the recall rate of our duplicate bug reports detection approach reaches about 95.75%, and the precision rate 90% or so which indicates the effectiveness of our LDA-based methods.

ACKNOWLEDGMENT

The work described in this paper was partially supported by Chongqing University Postgraduates' Innovation Project (Grant No. CYS15022), the National Natural Science Foundation of China (Grant no. 91118005, 61173131), and Changjiang Scholars and Innovative Research Team in University (Grant No. IRT1196).

REFERENCES

[1] J. Anvik, L. Hiew, and G. C. Murphy, "Coping with an open bug repository," in Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange - eclipse '05, 2005, pp. 35–39.

[2] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," in Software Engineering, 2007. ICSE 2007. 29th International Conference on, 2007, pp. 499–510.

[3] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language and execution information," in Proceedings of the 30th international conference on Software engineering, 2008, pp. 461–470.

[4] C. Sun, D. Lo, X. Wang, J. Jiang, and S.-C. Khoo, "A discriminative model approach for accurate duplicate bug report retrieval," in Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1, 2010, pp. 45–54.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," J. Mach. Learn. Res., vol. 3, pp. 993–1022, 2003.

[6] T. Hofmann, "Probabilistic latent semantic indexing," in Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, 1999, pp. 50–57.

[7] A. K. McCallum, "Mallet: A machine learning for language toolkit," 2002.

[8] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," Commun. ACM, vol. 18, no. 11, pp. 613–620, Nov. 1975.

[9] H. M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno, "Evaluation methods for topic models," in Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 1105–1112.

[10] S. W. Thomas, B. Adams, A. E. Hassan, and D. Blostein, "Studying software evolution using topic models," Sci. Comput. Program., vol. 80, pp. 457–479, 2014.

[11] L. Hiew, "Assisted detection of duplicate bug reports," The University Of British Columbia, 2006.

[12] A. Sureka and P. Jalote, "Detecting Duplicate Bug Report Using Character N-Gram-Based Features," in 2010 Asia Pacific Software Engineering Conference, 2010, pp. 366–374.

[13] C. Sun, D. Lo, S. Khoo, and J. Jiang, "Towards more accurate retrieval of duplicate bug reports," in 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), 2011, pp. 253–262.

[14] C.-Z. Yang, H.-H. Du, S.-S. Wu, and X. Chen, "Duplication Detection for Software Bug Reports Based on BM25 Term Weighting," in Technologies and Applications of Artificial Intelligence (TAAI), 2012 Conference on, 2012, pp. 33–38.

[15] N. Jalbert and W. Weimer, "Automated duplicate detection for bug tracking systems," in Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on, 2008, pp. 52–61.

[16] Y. Tian, C. Sun, and D. Lo, "Improved Duplicate Bug Report Identification," in 2012 16th European Conference on Software Maintenance and Reengineering, 2012, pp. 385–390..