ISO 9001 : 2008

TRƯỜNG ĐẠI HỌC TRÀ VINH

HỘI ĐỒNG KHOA HỌC

BÁO CÁO TỔNG KẾT

ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CẤP TRƯỜNG

CẢI TIẾN VIỆC THỰC THI DÒ TÌM NHỮNG BÁO CÁO LỖI TRÙNG NHAU SỬ DỤNG THÔNG TIN CENTROID CLASS MỞ RỘNG

Chủ nhiệm đề tài: **ThS. NHAN MINH PHÚC**

Chức danh: **Giảng viên**

Đơn vị: **Khoa Kỹ thuật và Công nghệ**

Trà Vinh, ngày 06 tháng 08 năm 2017

TRƯỜNG ĐẠI HỌC TRÀ VINH

HỘI ĐỒNG KHOA HỌC

ISO 9001 : 2008

BÁO CÁO TỔNG KẾT

ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CẤP TRƯỜNG

CẢI TIẾN VIỆC THỰC THI DÒ TÌM NHỮNG BÁO CÁO LỖI TRÙNG NHAU SỬ DỤNG THÔNG TIN CENTROID CLASS MỞ RỘNG

Xác nhận của cơ quan chủ quản

(Ký, đóng dấu, ghi rõ họ tên)

Chủ nhiệm đề tài

(Ký, ghi rõ họ tên)

Nhan Minh Phúc

Trà Vinh, ngày 06 tháng 08 năm 2017

TÓM TẮT

Trong việc bảo trì phần mềm, những báo cáo lỗi đóng một vai trò quan trọng đối với sự chính xác của những gói phần mềm. Thật không may, vấn đề báo cáo lỗi trùng nhau lại xảy ra, lý do có quá nhiều báo cáo lỗi được gửi đến trong những dự án phần mềm khác nhau, dẫn đến nhiều báo cáo lỗi bị trùng nhau, và việc xử lý thường tốn nhiều thời gian và chi phí trong vấn đề bảo trì phần mềm.Trong nghiên cứu này sẽ giới thiệu một phương pháp dò tìm dựa vào thông tin centroid lớp mở rộng (*Extended Class Centroid Information (ECCI)*) để cải tiến việc thực thi dò tìm. Phương pháp này được mở rộng từ phương pháp trước đây chỉ sử dụng centroid mà không xem xét đến những tác động của cả hai lớp bên trong là inner và inter. Ngoài ra phương pháp này cũng cải tiến việc sử dụng normalized cosine trước đây cho việc xác định sự giống nhau giữa hai báo cáo lỗi bằng denormalized cosine. Hiệu quả của phương pháp ECCI được minh chứng thông qua việc thực nghiệm với ba dự án mã nguồn mở là: SVN, Argo UML và Apache. Kết quả thực nghiệm cho thấy rằng phương pháp ECCI cho kết quả dò tìm tốt hơn những phương pháp khác khoảng 10% trong tất cả các trường hợp khi được so sánh.

In software maintenance, bug reports play an important role for the correctness of software packages. Unfortunately, a duplicate bug report problem arises because there are too many duplicate bug reports in various software projects. Processing duplicate bug reports is thus time-consuming and has high cost of software maintenance. In this research introduces a detection scheme based on the extended class centroid information (ECCI) to enhance the detection performance. This method is extended from the previous one which used only centroid method without considering the effects of both inner and inter class. Besides, this method also improved the use of normalized cosine previously for identifying the similarity between two bug reports by denormalized cosine.The effectiveness of ECCI is verified in an empirical study with three open-source projects, SVN, Argo UML, and Apache. The experimental results show that ECCI outperforms other detection schemes by about 10% in all cases.

MỤC LỤC

Nội dung **Trang**

DANH MỤC BẢNG BIỂU

DANH MỤC CÁC BIỂU ĐỒ, SƠ ĐỒ, HÌNH ẢNH

LỜI CẢM ƠN

Nghiên cứu này được tài trợ từ nguồn kinh phí Nghiên cứu Khoa học của Trường Đại học Trà Vinh. Tôi xin chân thành cảm ơn Ban Giám Hiệu, Phòng, Khoa, Bộ môn đã tạo điều kiện cho tôi thực hiện nghiên cứu này. Ngoài ra tôi cũng rất biết ơn những đồng nghiệp luôn ủng hộ và hỗ trợ cho tôi trong quá trình thực hiện nghiên cứu đề tài này.

PHẦN MỞ ĐẦU

1. Tính cấp thiết của đề tài

Hiện nay số lượng người sử dụng những kho phần mềm mã nguồn mở rất nhiều, trong quá trình sử dụng, nếu có phát sinh lỗi, người dùng sẽ gởi những thông báo lỗi này đến hệ thống xử lý lỗi, do số người sử dụng những phần mềm mã nguồn mở ngày càng nhiều nên số lỗi được gởi đến hệ thống xử lý lỗi cũng ngày càng lớn. Khi đó sẽ có những báo cáo lỗi do những người dùng gởi đến với cùng một nội dung lỗi, do đó dẫn đến nội dung báo cáo lỗi trùng nhau, với số lượng lớn báo cáo lỗi được gởi đến, sẽ rất khó khăn để xác định, những báo cáo lỗi nào trùng nhau để tránh việc xử lý lại những báo cáo lỗi đã xử lý rồi. Theo thống kê của Mozilla từ tháng 5/2003 đến tháng 4/2008 có đến 420.000 báo cáo lỗi do người dùng gởi, trong đó có tới 30% là những báo cáo lỗi bị trùng nhau. Tương tự với Eclipse, từ 10/2001 đến 4/2008 có 225.000 báo cáo lỗi được gởi bởi người dùng, trong đó 20% được thống kê là những báo cáo lỗi trùng nhau. Để lọc ra những báo cáo lỗi trùng nhau do người dùng gởi đến, theo như thống kế được báo cáo bởi Runeson, Alexandersson và Nyhol thì trung bình một người phải mất 30 phút để tìm ra một báo cáo lỗi trùng nhau, điều này làm lãng phí thời thời gian và công sức, và với số lượng ngày càng tăng từ những báo cáo lỗi gởi đến, việc xác định những báo cáo lỗi trùng nhau càng khó khăn hơn, mất nhiều thời gian và công sức hơn. Từ thực trạng này cho thấy tầm quan trọng của việc đưa ra những giải pháp trong việc xử lý những báo cáo lỗi trùng nhau là hết sức cần thiết và cấp bách, vì vậy việc nhận biết những báo cáo lỗi trùng nhau đóng vai trò rất quan trọng và mang lại nhiều lợi ích: thứ nhất, tiết kiệm được thời gian và công sức con người cho việc phân tích lỗi. Thứ hai, những thông tin chứa trong những báo cáo lỗi trùng nhau có thể rất hữu ích cho việc tìm, và xử lý lỗi, đó cũng chính là tính cấp thiết cần triển khai nghiên cứu khoa học cho đề tài này để xử lý những báo cáo lỗi trùng nhau trên những kho phần mềm mã nguồn mở.

2. Tổng quan nghiên cứu

2.1. Tình hình nghiên cứu trong nước

Ở Việt Nam những năm qua đã có nhiều công trình nghiên cứu về lĩnh vực xử lý ngôn ngữ tự nhiên, đặc biệt là xử lý ngôn ngữ tiếng Việt, tuy nhiên hầu hết những nghiên cứu trong nước chưa thấy có công trình nào nghiên cứu về vấn đề xử lý ngôn ngữ tiếng anh liên quan đến báo cáo lỗi, do vấn đề nghiên cứu này còn khá mới, được công bố trên các tạp chí ngoài nước từ 2005. Do đó

tình hình nghiên cứu trong nước còn hạn chế. Một số công trình nghiên cứu trong nước có liên quan đến lĩnh vực nghiên cứu này có thể liệt kê như sau:

1. Nguyễn Linh Giang, Nguyễn Mạnh Hiển, "Phân loại văn bản tiếng Việt với bộ phân loại vecto hỗ trợ SVM". Tạp chí CNTT&TT, tháng 6 năm 2006.

2. Nguyễn Linh Giang, Nguyễn Duy Hải, "Mô hình thống kê hình vị tiếng Việt và ứng dụng", Chuyên san "Các công trình nghiên cứu, triển khai Công nghệ Thông tin và Viễn thông, Tạp chí Buu chính Viễn thông, số 1, tháng 7-1999, trang 61-67. 1999.

3. Huỳnh Quyết Thắng, Đinh Thị Thu Phương, "Tiếp cận phương pháp học không giám sát trong học có giám sát với bài toán phân lớp văn bản tiếng Việt và đề xuất cải tiến công thức tính độ liên quan giữa hai văn bản trong mô hình vectơ", kỷ yếu Hội thảo ICT.rda'04, trang 251-261, Hà Nội 2005.

4. Đỗ Phúc, "Nghiên cứu ứng dụng tập phổ biến và luật kết hợp vào bài toán phân loại van bản tiếng Việt có xem xét ngữ nghĩa", Tạp chí phát triển KH&CN, tập 9, số 2, pp. 23-32, nam 2006 .

5. Trần Cao Đệ, Phạm Nguyên Khang, "phân loại văn bản với máy học Vector hỗ trợ và cây quyết định", trang 52-63, Tạp chí Khoa học, Đại học Cần Thơ.

Ngoài ra còn có một số công trình nghiên khác được nêu trong tài liệu tham khảo của quyển thuyết minh đề tài. Tuy nhiên chưa có đề tài nào đề cập đến việc nghiên cứu những báo cáo lỗi trùng nhau trong những kho phần mềm mở, do đó đây là một đề tài mới, chưa được nghiên cứu rộng rải tại Việt Nam.

2.2. Tình hình nghiên cứu ngoài nước

Trên thế giới cho tới nay có rất nhiều công trình nghiên cứu liên quan đến lĩnh vực này, cụ thể bắt đầu năm 2005, Lyndon Hiew đã công bố công trình nghiên cứu có tên "Coping with an Open Bug Repository" sử dụng xử lý ngôn ngữ tự nhiên cho kỹ thuật phân cụm tăng trưởng dựa vào centroid, kết quả đạt được trong việc dò tìm những báo cáo lỗi trùng nhau chính xác chiếm 30%-50%. Năm 2008, Wang et al. đã công bố công trình mang tên "*An Approach to Detecting Duplicate Bug Reports using Natural Language and Execution Informa tion* ", trong đó sử dụng kỹ thuật xử lý ngôn ngữ tự nhiên kết hợp với thông tin thực thi của người dùng, kết quả đạt được tỷ lệ dò tìm báo cáo lỗi trùng nhau chiếm 67%-93% for Firefox. Năm 2010, Sureka và Jalote công bố công trình mang tên "*Detecting Duplicate Bug Report using Character N* -

Gram-based Features", sử dụng data mining, với kỹ thuật n-gram cho kho phần mềm Eclipse, tuy nhiên kết quả đạt được tỷ lệ dò tìm những báo cáo lỗi trùng nhau chỉ chiếm 40%. Năm 2014, một công trình được công bố mang tên "Duplicate Bug Report Detection Using Clustering", sử dụng kỹ thuật phân cụm, phương pháp này đạt tỷ lệ chính xác là 24%.

Ngoài ra, còn rất nhiều công trình nghiên cứu khác liên quan đến lĩnh vực này được liệt kê một số bên dưới. Mỗi công trình nghiên cứu đều đưa ra những kỹ thuật, phương pháp và có cách tiếp cận khác nhau trong việc dò tìm những báo cáo lỗi trùng nhau. Tuy nhiên kết quả đạt được vẫn đang là một thách thức, điều này làm cho việc ứng dụng thực tế vẫn chưa đáp ứng được. Một số công trình có thể được liệt kê cụ thể bên dưới:

1. John Anvik, Lyndon Hiew, and Gail 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. Nicolas Bettenburg, Sascha Just, Adrian Schröter, Cathrin Weiss, Rahul Premraj, and Thomas Zimmermann, "What Makes a Good Bug Report?" in Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (SIGSOFT '08/FSE-16), 2008, pp. 308–318.
3. Nicolas Bettenburg, Rahul Premraj, Thomas Zimmermann, and Sunghun Kim, "Duplicate Bug Reports Considered Harmful... Really?" in Proceedings of the 24th IEEE International Conference on Software Maintenance (ICSM 2008), 2008, pp. 337–345.
4. Yguaratã Cerqueira Cavalcanti, Paulo Anselmo da Mota Silveira Neto, Euardo Santana de Almeida, Daniel Lucrédio, Carlos Eduardo Albuquerque da Cunha, and Silvio Romero de Lemos Meira, "One Step More to Understand the Bug Report Duplication Problem", in Proceedings of the 24th Brazilian Symposium on Software Engineering (SBES'10), 2010, pp. 148–157.
5. Yguaratã Cerqueira Cavalcanti, Eduardo Santana de Almeida, Carlos Eduardo Al buquerque da Cunha, Daniel Lucrédio, and Silvio Romero de Lemos Meira, "An Initial Study on the Bug Report Duplication Problem", in Proceedings of the 14th European Conference on Software Maintenance and Reengineering, 2010, pp. 264–267.

6. Zhi-Hao Chen, “Duplicate Detection on Bug Reports using N-Gram Features and Cluster Shrinkage”, Master Thesis, Yuan Ze University, Jul. 2011.
7. Hung-Hsueh Du, “A Study of Duplication Detection Methods for Bug Reports based on BM25 Feature Weighting,” Master Thesis, Yuan Ze University, Nov. 2011.
8. Hu Guan, Jingyu Zhou, and Minyi Guo, “A Class-Feature-Centroid Classifier for Text Categorization” in Proceedings of the 18th International Conference on World Wide Web (WWW 2009), 2009, pp. 201–210.
9. Eui-Hong Han and George Karypis, “Centroid-Based Document Classification: Analysis and Experimental Results,” in Proceedings of the Fourth European Con- ference on Principles of Data Mining and Knowledge Discovery (PKDD ’00), 2000, pp. 424–431.
10. Lyndon Hiew, “Assisted Detection of Duplicate Bug Reports,” Master Thesis, The University of British Columbia, May 2006.
11. Nicholas Jalbert and Westley Weimer, “Automated Duplicate Detection for Bug Tracking Systems,” in Proceedings of the 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2008), 2008, pp. 52–61.
12. Mostafa Keikha, Narjes Sharif Razavian, Farhad Oroumchian, and Hassan Seyed Razi, “Document Representation and Quality of Text: An Analysis,” in Survey of Text Mining II, Michael W. Berry and Malu Castellanos, Eds.Springer London, 2008, ch. 12, pp. 219–232.
13. Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford, “Okapi at TREC-3,” in Proceedings of the Third Text REtrieval Conference (TREC-3), 1994, pp. 109–126.
14. Per Runeson, Magnus Alexandersson, and Oskar Nyholm, “Detection of Duplicate Defect Reports using Natural Language Processing,” in Proceedings of the 29th In- ternational Conference on Software Engineering (ICSE 2007), 2007, pp. 499–510.
15. Chengnian Sun, David Lo, Xiaoyin Wang, Jing Jiang, and Siau-Cheng Khoo, “A Discriminative Model Approach for Accurate Duplicate Bug Report Retrieval,” in Proceedings of the 32nd ACM/IEEE International Conference on Software Engi- neering (ICSE 2010), vol. 1, 2010, pp. 45–

54.

16. Ashish Sureka and Pankaj Jalote, “Detecting Duplicate Bug Report using Character N -Gram-based Features,” in Proceedings of the 17th Asia Pacific Software Engi- neering Conference (APSEC 2010), 2010, pp. 366–374.

17. Xiaoyin Wang, Lu Zhang, Tao Xie, John Anvik, and Jiasu Sun, “An Approach to Detecting Duplicate Bug Reports using Natural Language and Execution Informa tion,” in Proceedings of the 30th International Conference on Software Engineering (ICSE ’08), 2008, pp. 461–470.

18. Xiaoyan Zhang, Ting Wang, Xiaobo Liang, Feng Ao, and Yan Li, “A Class-based Feature Weighting Method for Text Classification,” Journal of Computational Infor- mation System, vol. 3, pp. 965–972, 2012.

19. Vincent Boisselle, Bram Adams MCIS, Polytechnique Montreal, Québec, “The Impact of Cross-Distribution Bug Duplicates, Empirical Study on Debian and Ubuntu”, IEEE 15th International Working Conference on Source Code Analysis and Manipulation (SCAM), 2015, page 131-140.

20. Chao-Yuan Lee, Dan-Dan Hu, Zhong-Yi Feng, Cheng-Zen Yang, "Mining Temporal Information to Improve Duplication Detection on Bug Reports", Advanced Applied Informatics (IIAI-AAI) 2015 IIAI 4th International Congress on, pp. 551-555, 2015.

Trong đề tài nghiên cứu “cải tiến việc thực thi dò tìm những báo cáo lỗi trùng nhau sử dụng thông tin centroid class mở rộng”, giới thiệu phương pháp mới trong đó nghiên cứu phương pháp và đề xuất kỹ thuật xử lý mới nhằm cải thiện dò tìm những báo cáo lỗi trùng nhau hiệu quả hơn.

3. Mục tiêu

- **Mục tiêu chung**

Đối với nhiều dự án mã nguồn mở, số lỗi báo cáo trùng nhau chiếm một số lượng đáng kể trong kho chứa lỗi, vì vậy việc nhận biết tự động những báo cáo lỗi trùng nhau đóng vai trò quan trọng và cần thiết, giúp tiết kiệm thời gian và công sức cho con người. Mục tiêu của nghiên cứu này giới thiệu một phương pháp mới nhằm cải tiến việc thực thi dò tìm những báo cáo lỗi trùng nhau với độ chính xác tốt hơn so với những phương pháp nghiên cứu đã được công bố trước đó. Trong nghiên cứu này chúng tôi sử dụng thông tin centroid class mở rộng và thực nghiệm trên ba dự án phần mềm mã nguồn mở là

Apache, ArgoUML và SVN. Kết quả thực nghiệm chúng tôi muốn rằng, phương pháp được giới thiệu có thể hiệu quả cải tiến việc thực thi dò tìm những báo cáo lỗi trùng nhau với tỷ lệ chính xác tốt hơn so với những phương pháp được công bố trước đây.

- **Mục tiêu cụ thể 1**
    1. Trích ra được những đặc điểm cần thiết từ những file báo cáo
    2. Tiền xử lý sử dụng kỹ thuật ngôn ngữ tự nhiên loại bỏ những từ dư thừa, những từ không có nghĩa, những dấu câu không cần thiết trong file báo cáo lỗi.
- **Mục tiêu cụ thể 2**
    1. Phân cụm cho các báo cáo lỗi
    2. Tính trọng lượng đặc điểm class của mỗi từ trong các file báo cáo lỗi
    3. Sử dụng mô hình không gian vector
- **Mục tiêu cụ thể 3**
    1. Tính thông tin centroid class mở rộng
    2. Tính độ giống nhau giữa các file báo cáo lỗi
    3. Sắp xếp mức độ giống nhau nhất từ mức cao đến mức thấp trong top 20 của những file báo cáo lỗi trùng nhau.
    4. Tính tỷ lệ chính xác dò tìm báo cáo lỗi trùng nhau của phương pháp được giới thiệu.

4. Đối tượng, phạm vi và phương pháp nghiên cứu

Đối với đề tài này, đối tượng nghiên cứu là những hệ thống chứa kho phần mềm mở như Firefox, Eclipse, Apache, SVN...tuy nhiên do số lượng hệ thống nhiều và mỗi hệ thống có một vài đặc điểm quản lý khác nhau, do đó nghiên cứu này chỉ thực hiện đối với 3 kho phần mềm mở là SVN, Argo UML, và Apache.

5. Đối tượng, địa điểm và thời gian nghiên cứu

Đối tượng nghiên cứu 3 kho phần mềm mở là SVN, Argo UML, và Apache với thời gian nghiên cứu 9 tháng.

a. Quy mô nghiên cứu

Đề tài này nghiên cứu tập trung vào các kho phần mềm mã nguồn mở có hỗ trợ hệ thống quản lý lỗi, cụ thể với với 3 kho phần mềm mở là SVN, Argo UML, và Apache. Dataset được tiến hành thực nghiệm nghiên cứu trên 2000 file báo cáo lỗi đối với hệ thống SVN, trên 2700 file báo cáo lỗi với hệ thống Argo UML, cuối cùng trên 4500 file báo cáo lỗi đối kho phần mềm Argo UML.

b. Phương pháp nghiên cứu

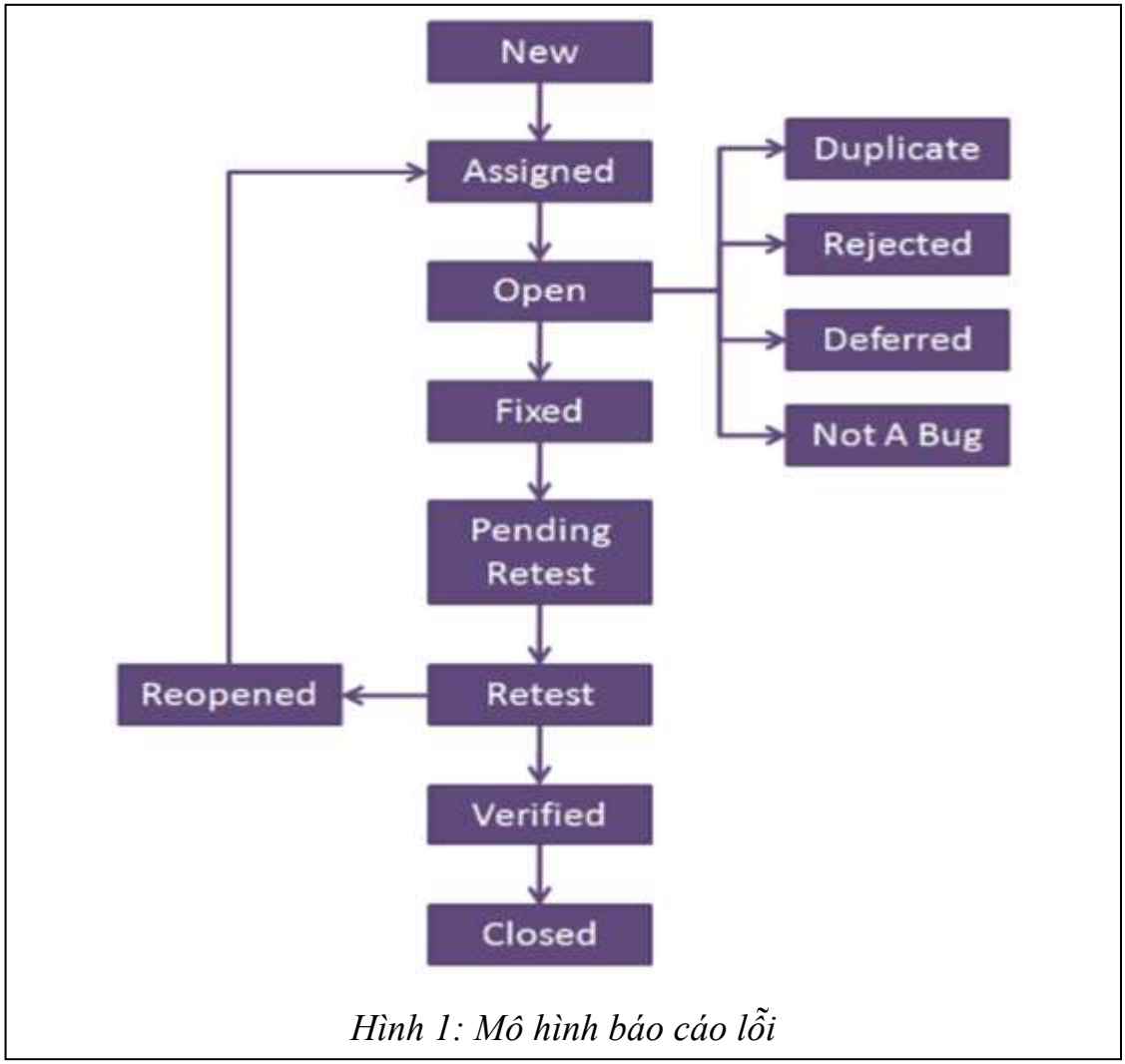
- Nghiên cứu tài liệu từ các bài báo, các công trình đã được công bố trên các tạp chí uy tín về lĩnh vực nghiên cứu liên quan.
- Download dữ liệu cho việc thực nghiệm các file báo cáo lỗi từ trang web của kho phần mềm mở là SVN, ArgoUML, Apache.
- Xây dựng demo lại một vài phương pháp đã công bố trên các tạp chí uy tín.
- Nghiên cứu và xây dựng phương pháp dò tìm sử dụng centroid class mở rộng.
- Đánh giá kết quả và so sánh với các phương pháp đã được công bố.

PHẦN NỘI DUNG

CHƯƠNG 1: QUY TRÌNH XỬ LÝ LỖI VÀ RÚT TRÍCH ĐẶC ĐIỂM TỪ FILE BÁO CÁO LỖI

1. Quy trình xử lý báo cáo lỗi

Quy trình báo cáo lỗi được thực hiện như hình 1. Khi một báo cáo lỗi vừa được gửi đến, nó sẽ được gắn trạng thái *"New"*. Sau đó lỗi sẽ được bộ phận kiểm tra lỗi (tester) kiểm tra, nếu đây là lỗi thật sẽ được giao cho một lập trình viên tương ứng để xử lý, khi đó trạng thái báo cáo lỗi sẽ là *"Assigned"*. Trạng thái *"Open"* là khi lập trình viên bắt đầu phân tích và tiến hành xử lý lỗi. Nếu

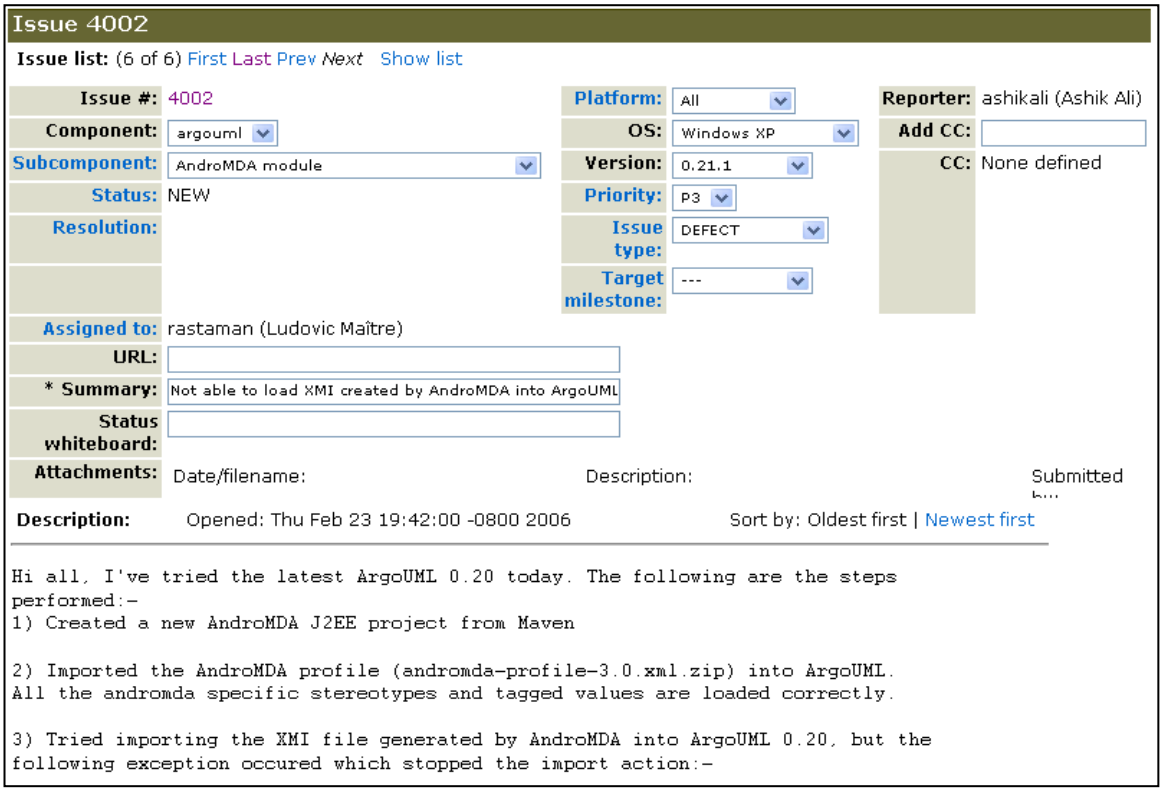


Hình 1: Mô hình báo cáo lỗi

quá trình kiểm tra phát hiện báo cáo lỗi này đã được báo trước đó rồi, khi đó gán trạng thái là *"Duplicate"*. Trạng thái *"Rejected"* được gán nhãn khi tester phát hiện lỗi này không có thật.Nếu báo cáo lỗi mà khi xử lý lỗi liên quan đến quá nhiều yếu tố có thể ảnh hưởng đến phần mềm, khi đó lỗi này sẽ được sửa trong phiên bản sau và báo cáo lỗi được dán nhãn *"Deferred"*. Trạng thái *"Not a bug"* được gán khi tester phát hiện lỗi này không phải là một lỗi phần

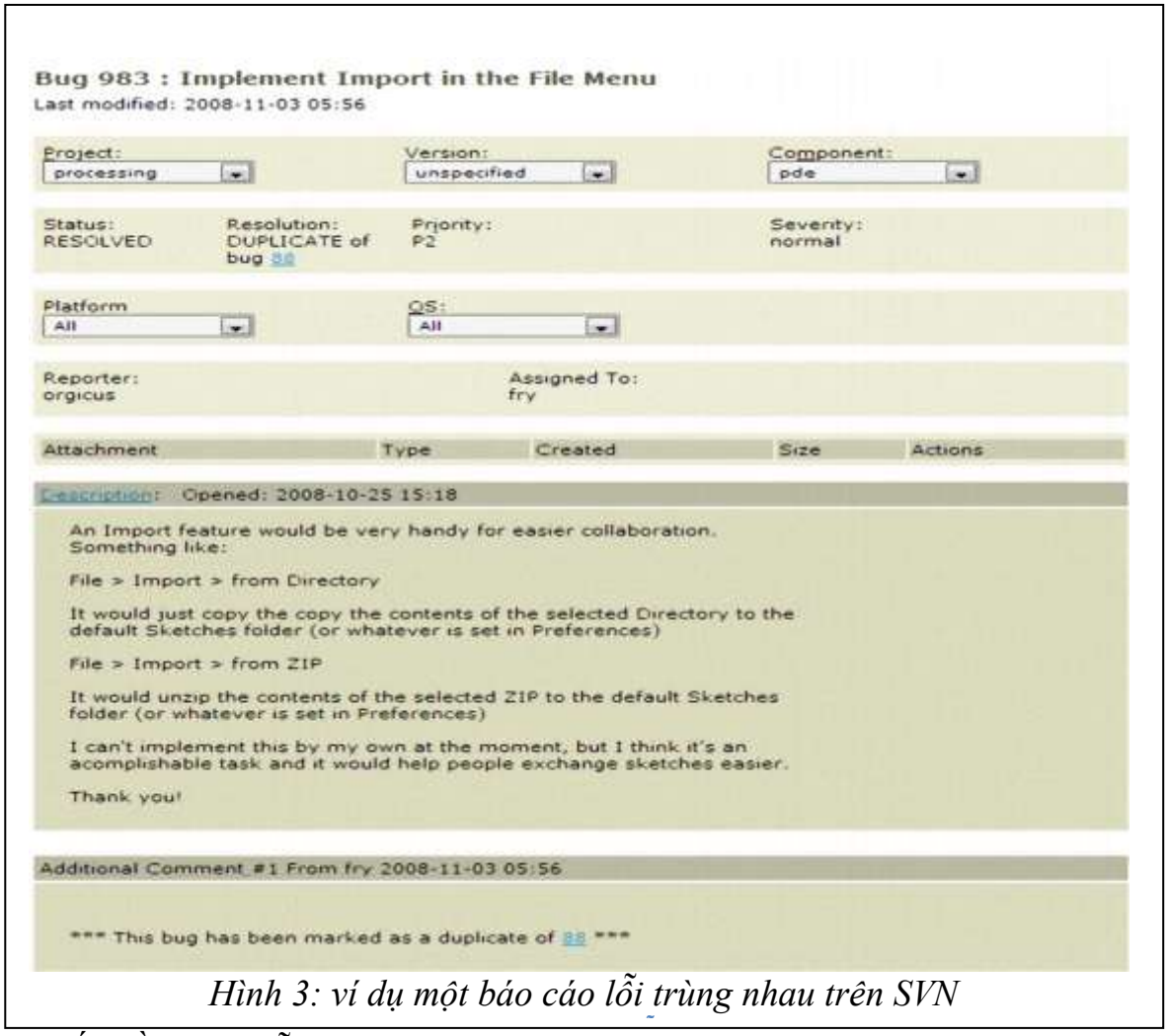
mềm mà thuộc chức năng phần mềm không hỗ trợ. Trạng thái *"Fixed"* được gán khi lập trình viên đã xử lý xong lỗi và chuyển đến bộ phận kiểm tra lỗi để kiểm tra lại. *"Pending retest"* là trạng thái mà báo cáo lỗi đang trong quá trình kiểm tra lại. *"Retest"* là trạng thái báo cáo lỗi được kiểm tra lại để biết lỗi đã sửa xong hay chưa. Nếu tester phát hiện vẫn còn lỗi, khi đó báo cáo lỗi sẽ được gán "*Reopen"*, khi đó báo cáo lỗi này sẽ được xử lý lại. Nếu tester xác nhận báo cáo này đã được sửa xong, khi đó sẽ được gán nhãn *"Closed"*.

Theo tìm hiểu trong những năm gần đây, tình hình nghiên cứu về báo cáo lỗi trùng nhau trong các kho phần mềm mở tại Việt Nam còn rất hạn chế và hầu như chưa có, hầu hết những nghiên cứu chỉ tập trung ở nước ngoài. Tuy nhiên phần lớn phương pháp họ sử dụng mô hình không gian vector (Vector Space Model) kết hợp với việc tính độ giống nhau giữa hai báo cáo lỗi [1, 2, 3, 4, 10, 11].Gần đây phương pháp xử lý ngôn ngữ tự nhiên [9] đã được giới thiệu, phương pháp này được thực hiện kết hợp với thông tin thực thi của báo cáo lỗi, mặc dù kết quả cho thấy có sự cải thiện trong việc dò tìm lỗi trùng nhau so với những phương pháp trước, nhưng hiệu quả vẫn còn khá hạn chế. Chính vì điều này, phương pháp ECCI được giới thiệu với việc sử dụng xử lý ngôn ngữ tự nhiên cơ bản kết hợp với centroid class để tăng độ chính xác trong việc



Hình 2: Ví dụ về các thông tin trong một báo cáo lỗi

dò tìm những báo cáo lỗi trùng nhau, do phương pháp này xem xét đến những tác động của cả hai lớp bên trong là inner và inter. Kết quả thực nghiệm đã cho thấy phương pháp này có sự cải tiến đáng kể so với những phương pháp trước đây.



Hình 3: ví dụ một báo cáo lỗi trùng nhau trên SVN

2. Vấn đề dò tìm lỗi trùng nhau

Khi người dùng sử dụng phần mềm mà phát sinh lỗi, thông tin báo cáo lỗi khi đó sẽ được gởi đến hệ thống quản lý phần mềm tương ứng. Một thông tin báo cáo lỗi là một dữ liệu có cấu trúc bao gồm nhiều trường như: tóm tắt lỗi (summary), mô tả lỗi (description), hệ điều hành sử dụng (OS)…như trong hình 2. Trường tóm tắt lỗi thường là những mô tả ngắn gọn về vấn đề lỗi phát sinh, trong khi đó trường mô tả lỗi thường được xem là quan trọng nhất, lý do trường này mô tả chi tiết về lỗi phát sinh cũng như thao tác người dùng thực hiện gây ra lỗi.Trường hệ điều hành sẽ cho biết thông tin hệ điều hành của người dùng khi sử dụng phần mềm gây ra lỗi, điều này cũng giúp dễ dàng hơn cho lập trình viên trong việc khắc phục lỗi phần

mềm. Ngoài ra nó cũng có phần bình luận cho những người báo cáo lỗi khác bình luận. Nếu một báo cáo lỗi là báo cáo đầu tiên, nó được gọi là báo cáo lỗi chính (master bug report). Ngược lại, nó sẽ được gán lỗi trùng nhau sau khi được xử lý kiểm tra giống báo cáo lỗi chính.Trong hình 3, báo cáo lỗi có mã số *983* được thông báo trùng với báo cáo lỗi trước đó có mã số *88*.Để dò tìm những báo cáo lỗi trùng nhau, đầu tiên chúng ta phải rút trích những thông tin văn bản từ những báo cáo lỗi. Thông thường, một báo cáo lỗi bao gồm nhiều thông tin như nội dung tóm tắt lỗi, phần mô tả lỗi, hệ điều hành…Ví dụ bên dưới cho thấy thông tin sau khi rút trích ra file text của báo cáo lỗi 983.

Description: Opened: 2008-10-25 15:18
An Import feature would be very handy for easier collaboration. Something like:

File > Import > from Directory

It would just copy the copy the contents of the selected Directory to the default Sketches folder (or whatever is set in Preferences)

File > Import > from ZIP

It would unzip the contents of the selected ZIP to the default Sketches folder (or whatever is set in Preferences)

I can't implement this by my own at the moment, but I think it's an acomplishable task and it would help people exchange sketches easier.

Thank you!
Additional Comment #1 From fry 2008-11-03 05:56

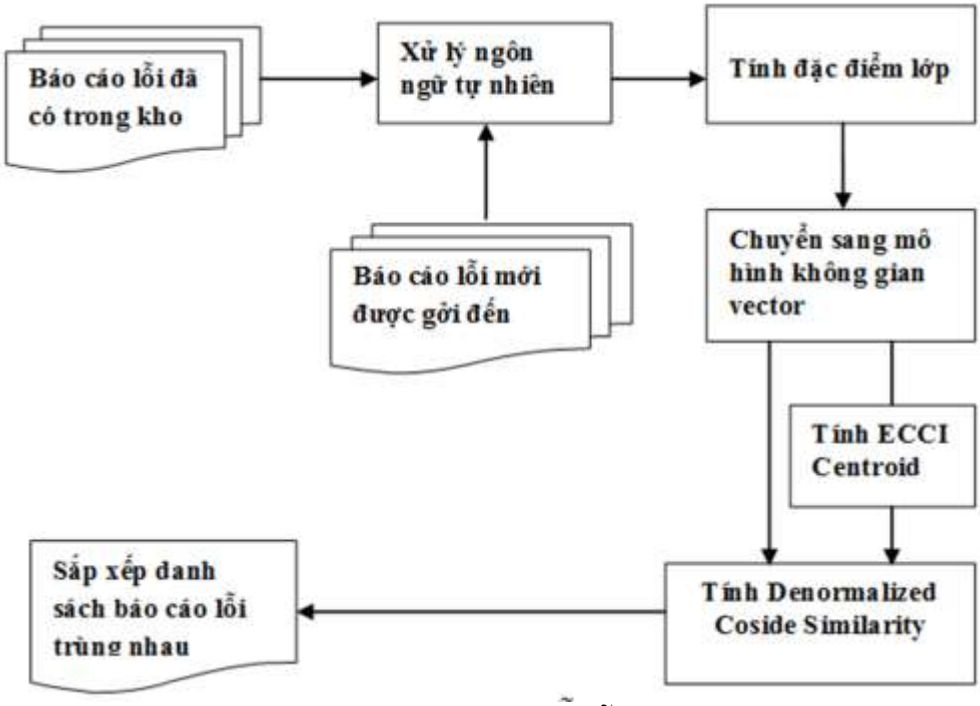
****** This bug has been marked as a duplicate of 88 ******

CHƯƠNG 2: PHƯƠNG PHÁP DÒ TÌM BÁO CÁO LỖI

2. Phương pháp dò tìm lỗi trùng nhau

2.1. Tổng quan về xử lý dò tìm lỗi

Để xác định chẳng hay một báo cáo lỗi vừa được người dùng gửi đến có trùng với những báo cáo lỗi đã được gửi trước đây hay không với phương pháp ECCI. Phương pháp này được kế thừa và cải tiến từ phương pháp sử dụng đặc điểm lớp trong centroid [6], trong đó xem xét cả hai đặc điểm trọng lượng bên



Hình 4: Mô hình xử lý báo cáo lỗi trùng nhau theo ECCI

tronglớp để cải thiện cho việc phân loại báo cáo lỗi, cũng như xem xét thông tin lớp liên quan đến trọng lượng từ. Trong nghiên cứu này, một lớp được định nghĩa như một một cụm báo cáo lỗi trùng nhau. Trong tập dữ liệu, việc xem xét báo cáo lỗi trùng nhau dựa vào thông tin được đánh dấu trong báo cáo lỗi có dạng "*This bug has been marked as a duplicate of <bug report ID>*" như ví dụ trong hình 3.Khi đó thông tin centroid có thể được trích ra từ mỗi cụm để tính sự giống nhau giữa các báo cáo lỗi. Toàn bộ quy trình xử lý báo cáo lỗi trùng nhau theo phương pháp ECCI được thực hiện như sau:

1. Xử lý ngôn ngữ tự nhiên
2. Tính trọng lượng đặc điểm lớp trong báo cáo lỗi
3. Tính ECCI centroid
4. Tính sự giống nhau giữa các báo cáo lỗi sử dụng Denormalized Cosine
5. Sắp xếp các báo cáo lỗi trùng nhau

Hình 4 cho thấy toàn bộ quy trình xử lý báo cáo lỗi trùng nhau theo phương pháp ECCI, bao gồm năm bước, các bước thực hiện sẽ được mô tả chi tiết bên dưới.

2.2. Xử lý ngôn ngữ tự nhiên

Như hình 2 và hình 3, nội dung báo cáo lỗi ngoài những thông tin hữu ích mô tả lỗi, nó còn chứa nhiều thông tin mà nó không thật sự có íchcho việc tự động dò tìm lỗi trùng nhau, ví dụ những từ như "and, or, not, but, very…" hay những dấu câu như dấu gạch ngang, dấu ngoặc đơn…, vì vậy việc loại bỏ những từ không cần thiết này thì rất quan trọng, ảnh hưởng nhiều đến sự chính xác của các phương pháp dò tìm. Trong bước này, mỗi báo cáo lỗi sẽ được rút trích thông tin từhai trường chính trong báo cáo lỗi gồm trường tóm tắt lỗi (summary), mô tả lỗi (description), do các thông tin từ hai trường này mô tả đầy đủ và có nghĩa để hỗ trợ việc xử lý lỗi.Sau đó thông tin này sẽ được xử lý thông qua các bước xử lý ngôn ngữ tự nhiên ở mức cơ bản gồm tách từ(tokenization), tiếp theo là loại bỏ những từ không có nghĩa (stop words) ví dụ như như những từ "the,and, or,…",sau đó tiến hành chuyển tất cả các dạng biến thể của một từ trở về từ gốc (stemming).Những thao tác xử lý ngôn ngữ tự nhiên cơ bản này được hỗ trợ bởi công cụ hỗ trợ WVTool (Word Vector Tool). Công cụ này giúp việc xử lý các thao tác xử lý ngôn ngữ tự nhiên nhanh và dễ dàng hơn.

2.3. Tính trọng lượng đặc điểm lớp trong báo cáo lỗi

2.3.1 Tăng cường mở rộng thông tin lớp

Trong quy trình xử lý báo cáo lỗi, việc tính đặc điểm trọng lượng lớp vô cùng quan trọng, nó ảnh hưởng trực tiếp đến kết quả xác định sự giống nhau giữa hai báo cáo lỗi. Mỗi từ trong các báo cáo lỗi sẽ được xác định và chuyển sang mô hình không gian vector tương ứng với một trọng lượng. Phương pháp ECCI được thừa kế và cải tiến từClass-Feature-Centroid(CFC)[5, 6], và trọng lượng đặc điểm lớp [8]. Trong CFC, trọng lượng của từ w_{ij} được tính như sau:

$$w_{ij} = b^{\frac{DF_{t_i}^{j}}{|C_j|}} \times \log(\frac{|C|}{CF_{t_i}}) \qquad 3.1)$$

Trong đó t_i là từ (term) trong báo cáo lỗi, $DF_{t_i}^{j}$là số báo cáo lỗi chứa t_i của lớp C_j,$|C_j|$ là số báo cáo lỗi trong lớp C_j, $|C|$ là tổng số lớp, CF_{t_i}là số lớp chứa t_i, và b là tham số lớn hơn một, dùng để điều chỉnh cho trọng lượng w_{ij}. Trong

CFC, $b^{\frac{DF_{t_i}^j}{|C_j|}}$ xem xét đến số báo cáo lỗi chứa mức độ xuất hiện thường xuyên của một từ bên trong lớp.Công thức log xem xét mức độ giống như IDF (inverse document frequency) truyền thống.ECCI được cải tiến từ CFC và trên cơ sở dựa vào [5], khi đó mức độ thường xuyên của một từ tf_{ijk}của t_i trong báo cáo lỗi d_k, thuộc lớp C_jđược tính như sau:

$$tf_{ijk} = \frac{fre(t_i)}{fre(t_i) + d + h \times \frac{dl}{dl_{avg}}} \qquad (3.2)$$

Trong đó $fre(t_i)$ là số lần xuất hiện của $t_i$trong báo cáo lỗi $d_k$hoặc của lớp C_j, d là tham số điều chỉnh tránh cho mẫu số bằng 0, h là tham số ảnh hưởng đến chiều dài của báo cáo lỗi, dl là chiều dài của báo cáo lỗi d_k hoặc tổng chiều dài của trong lớp C_j, dl_{avg} là trung bình của chiều dài các báo cáo lỗi. Nếu $t_i \in d_K$, khi đó dl_{avg} được tính như sau:

$$\boldsymbol{dl_{avg} = \frac{\sum_{d_m \in C} dl(d_m)}{\sum_{C_n \in C} |C_n|}} \qquad (\mathbf{3.3})$$

Trong đó $|C_n|$là số báo cáo lỗi trong Cn Nếu $t_i \in C_j$ nhưng $t_i \notin d_K$, khi đó :

$$\boldsymbol{dl_{avg} = \frac{\sum_{d_m \in C} dl(d_m)}{|C|}} \qquad (\mathbf{3.4})$$

Trong đó |C| là tổng số lớp,d và h là hai tham số, và nó có thể nằm trong một khoảng giá trị tùy theo tập dữ liệu.Tuy nhiên trong nghiên cứu này chỉ xác định $0.3 \leq d \leq 0.8$ và $1.5 \leq h \leq 20.0$ để tìm ra giá trị tốt nhất cho d và h.

Bảng 1: các công thức tính trọng lượng bên trong lớp inner

Tên công thức	Chức năng
EXP-DF (CFC)	$I_{inner}^i = b^{\frac{DF_{t_i}^j}{\|C_j\|}}$
TF	$I_{inner}^i = tf_{ijk}$
EXP-TF	$I_{inner}^i = b^{tf_{ijk}}$
EXP-TF-DF	$I_{inner}^i = b^{tf_{ijk} \times \frac{DF_{t_i}^j}{\|C_j\|}}$

a) Chỉ số tác động bên t rong lớp Inner

Với việc mở rộng thông tin dựa vào lớp, khi đó bốn công thức để tính chỉ số tác động bên trong lớp Inner được giới thiệu, và được tiến hành thực nghiệm để tìm ra một công thức tốt nhất. Bảng 1

cho thấy bốn công thức dùng để tính trọng lượng bên trong lớp Inner.

b)Chỉ số tác động bên trong lớp Inter

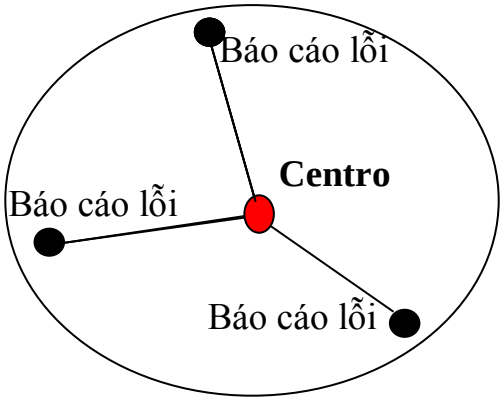
Để tăng cường độ chính xác trong việc phân loại báo cáo lỗi đối với chỉ số bên trong lớp I_{inter}, trong trường hợp này sử dụng theo phương pháp CFC:

$$I_{inter}^{i} = \log\left(\frac{|C|}{CF_{t_i}}\right) \quad (3.5)$$

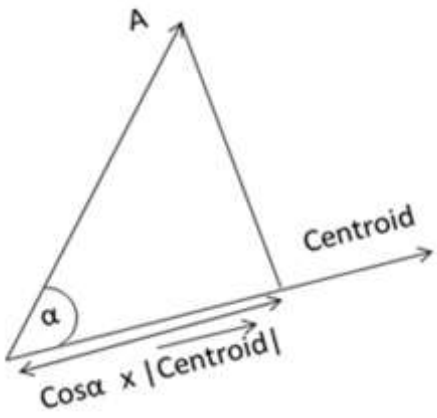
Nếu từ t_i xuất hiện trong tất cả các lớp, khi đó $I_{inter}^{i} = 0$, do $|C| = CF_{t_i}$, Nếu từ $t_i$xuất hiện chỉ trong một lớp, khi đó $I_{inter}^{i} = log\,|C|$. Trong trường hợp này, t_i có sự phân biệt tốt nhất trong các lớp báocáo lỗi trùng nhau.

2.4. Centroids vàECCI centroids

Phương pháp trong [2] sử dụng mô hình không gian vector cho cụm báo cáo lỗi của centroid. Trong phương pháp này, những báo cáo lỗi trùng nhau của cùng một nhóm thì được xem như một cụm, và vector centroid chính là trung bình cộng của các báo cáo lỗi trong cùng nhóm này như trong hình 5,khi đó nó được xem như là một báo cáo lỗi mới, đều này có nghĩa là khi một báo cáo mới được gửi đến, nó sẽ được so sánh với vector centroid của những cụm đã có trong kho lỗi thay cho việc so sánh với từng báo cáo lỗi. Trong khi đó centroid mở rộng sử dụng trong phương pháp ECCI cũng sử dụng giống centroid này, tuy nhiên điểm khác biệt là nó sử dụng lớp, trong đó xem xét đến các lớp inner và inter như đã đề cập phần 3.1.2 và 3.1.3 bên trong cùng một centroid. Điều này giúp cải thiện được việc so sánh chính xác hơn giữa hai báo cáo lỗi. ECCI centroids (EC) là một trong những thành phần quan trọng hỗ trợ việc tìm ra sự giống nhau giữa các báo cáo lỗi, nó là trung bình cộng của các vector báo cáo lỗi trong cùng một lớp C_j:



Hình 5: Mô hình centroid



Hình 6: Denormalize cosine

$$\overrightarrow{EC_j} = \frac{1}{|C_j|} \sum_{\vec{d}_k \in C_j} \vec{d}_k \qquad (3.6)$$

2.5. Tính sự giống nhau giữa các báo cáo lỗi với denormalized cosine

Trong bước này sẽ tiến hành xác định sự giống nhau giữa các báo cáo lỗi, khi có một báo cáo lỗi mới được gửi đến, nó sẽ được tính toán để xác định chẳng hay nó có trùng lắp với những báo cáo đã tồn tại trước đó hay chưa. Phương pháp truyền thống sử dụng cosise similaritytruyền thống như sau:

$$Sim\left(\vec{d}_a, \vec{d}_b\right) = \frac{\vec{d}_a \,.\, \vec{d}_b}{\left|\vec{d}_a\right| . \left|\overrightarrow{d}_b\right|} \qquad (3.7)$$

Tuy nhiên, với cosine similarity nó không xem xét sự tác động của dữ liệu $\vec{d}_b$, mà chỉ cho thấy sự khác nhau giữa hai báo cáo lỗi $\vec{d}_a$và $\vec{d}_b$, Vì vậyECCI sử dụng denormalized cosine [5] để xem xét trong những thay đổi của centroid khác nhau trong các lớp, điều này giúp cải thiện việc dò tìm trùng nhau trong các báo cáo lỗi. Hình 6 cho thấy cách tính sử dụng denormalize cosine.

2.6. Sắp xếp các báo cáo lỗi trùng nhau

Khi có những báo cáo mới được gửi đến, những báo cáo này sẽ được thực hiện kiểm tra và so sánh xem có trùng với những báo cáo đã được gởi trước đó không? Do phương pháp ECCI sử dụng thông tin centroid lớp mở rộng, khi đó những báo cáo lỗi được gửi đến sẽ được tính và sắp xếp theo giá trị giống nhau nhất từ cao xuống thấp theo danh sách top 20.

CHƯƠNG 3: KẾT QUẢ THỰC NGHIỆM

3.1. Môi trường thực nghiệm

Bảng 2: Thông tin về datasets của 3 dự án nguồn mở

Mô tả	ArgoUML	Apache	SVN
Ngôn ngữ	Java	C	C
Loại phần mềm	UML Tool	HTTP Server	SCM tool
Kho chứa lỗi	Tigris	Bugzilla	Tigris
Thời gian thu thập	02/2000-05/2007	01/2001-02/2007	03/2001-05/2007
Số báo cáo lỗi	4,613	2,771	2,296
Số báo cáo lỗi trùng	755	614	313

Phương pháp ECCI đã tiến hành thực nghiệm với 3 kho báo cáo lỗi của những dự án phần mềm mở là Argo UML, Apache, và SVN. Thống kê chi tiết về 3 kho phần mềm này được mô tả trong bảng 2.Để đánh giá phương pháp dò tìm ECCI, khi đó đơn vị đo lường gọi là *recall rate* được sử dụng, nó được tính dựa trên bao nhiêu báo cáo lỗi có thểđược dò tìm đúng trong danh sách những báo cáo lỗi trùng nhau, phương pháp này được sử dụng phổ biến trong việc đánh giá kết quả tìm báo cáo lỗi trùng nhau, và nó được định nghĩa như sau:

$$Recall\ rate = \frac{\text{Số những dựđoán đúng}}{\text{Tổng số những báo cáo lỗi trùng nhau}} \qquad (4.1)$$

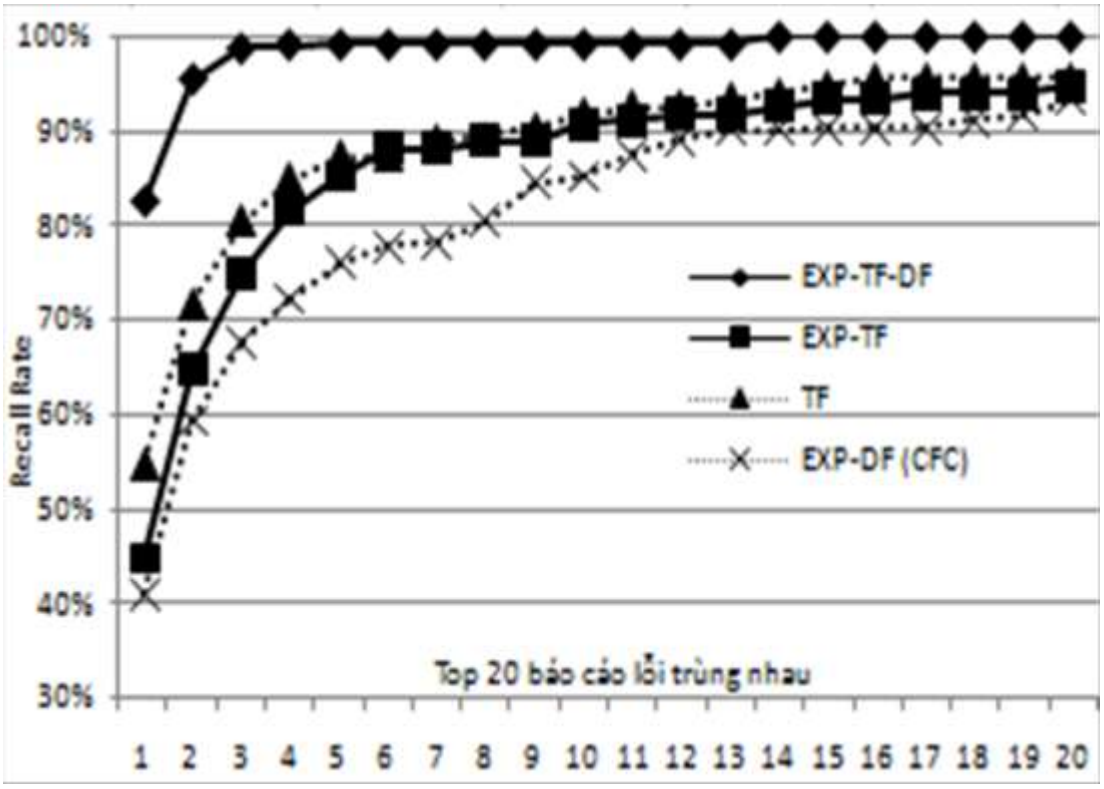
3.2. Những nhân tố tác động đến phương pháp ECCI

Trong phần nàymuốn thảo luận những nhân tố ảnh hưởng đến phương pháp ECCI. Thứ nhất là công thức được chọn cho việc tính trọng lượng đặc điểm lớp. Thứ hai và thứ ba liên quan đến việc xác định giá trị tốt nhất cho các tham số *b, d*, và *h*. Cuối cùng là công thức tính sự giống nhau giữa hai báo cáo lỗi sử dụng denormalized cosine.

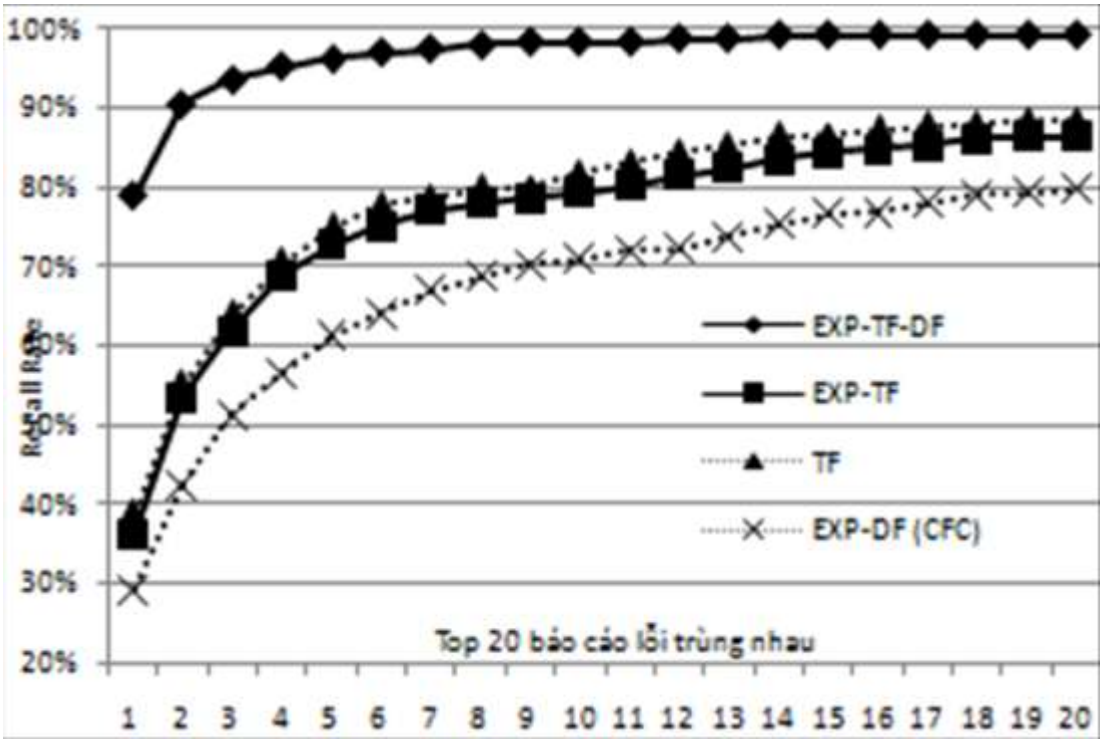
1. Trọng lượng đặc điểm lớp

Trong phần trước đã giới thiệu bốn công thức khác nhau cho việc tính trọng lượng từng từ, trong những báo cáo lỗi dựa vào lớp bao gồm: EXP-DF,TF,EXP-TF, và EXP-TF-DF.Khi tiến hành thực nghiệm để tìm ra công thức tốt nhất cho việc tính trọng lượng đặc điểm lớp,qua quan sát kết quả thực nghiệm cho thấy rằng EXP-TF-DF cho kết quả tốt nhất trong bốn công thức.

Lý do EXP-TF-DF cho kết quả tốt nhất có thể giải thích là do hỗ trợ nhiều cho thông tin từ dựa vào lớp, điều này cũng giải thích lý do CFC cho kết quả không tốt bởi nó đã không xem xét sự tác động những từ thương xuyên xuất hiện trong báo cáo lỗi dựa vào lớp. Hình 7 cho thấy sự vượt trội của phương

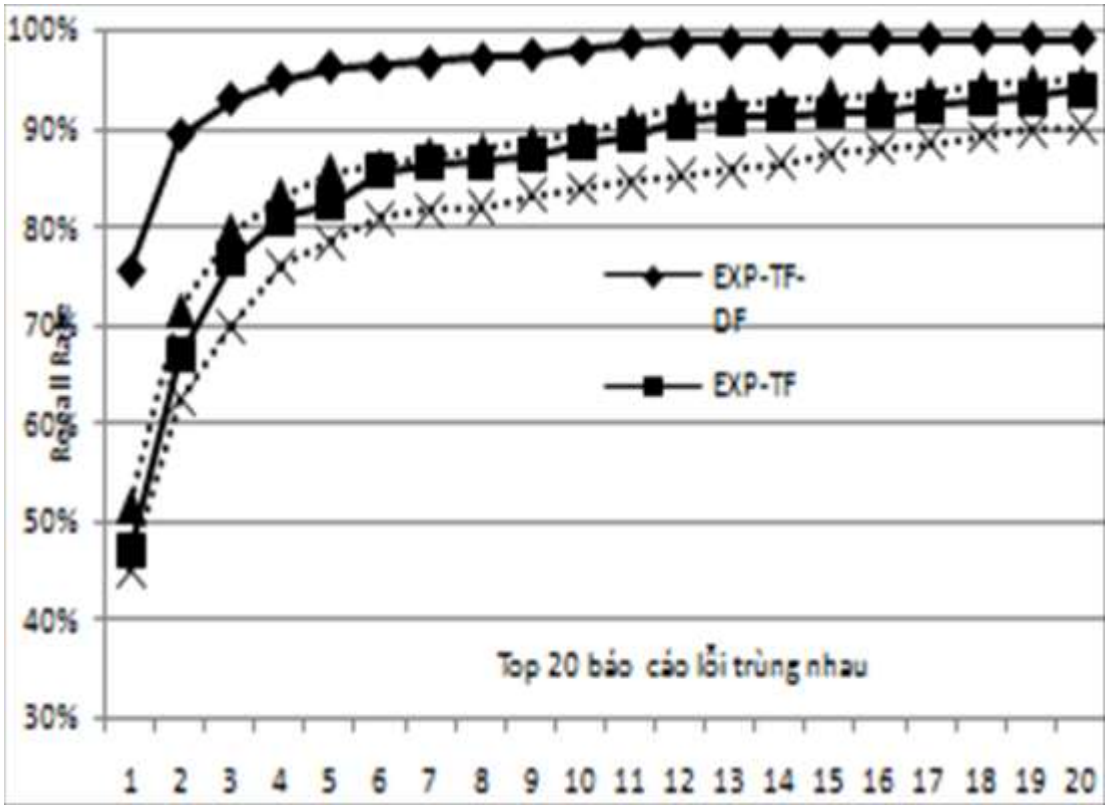


a) *Kết quả thực nghiệm trên SVN*



b) Kết quả thực nghiệm trên Argo UML

pháp EXP-TF-DF.

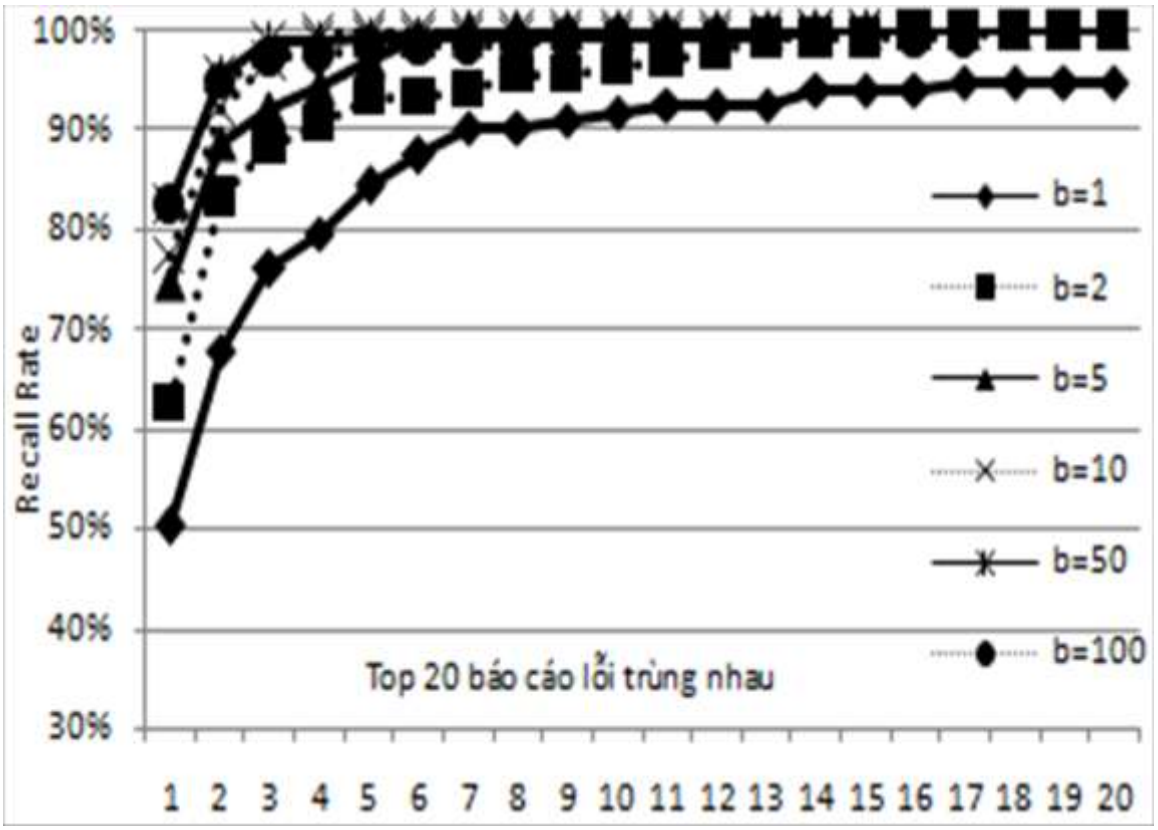


c) Kết quả thực nghiệm trên Apache

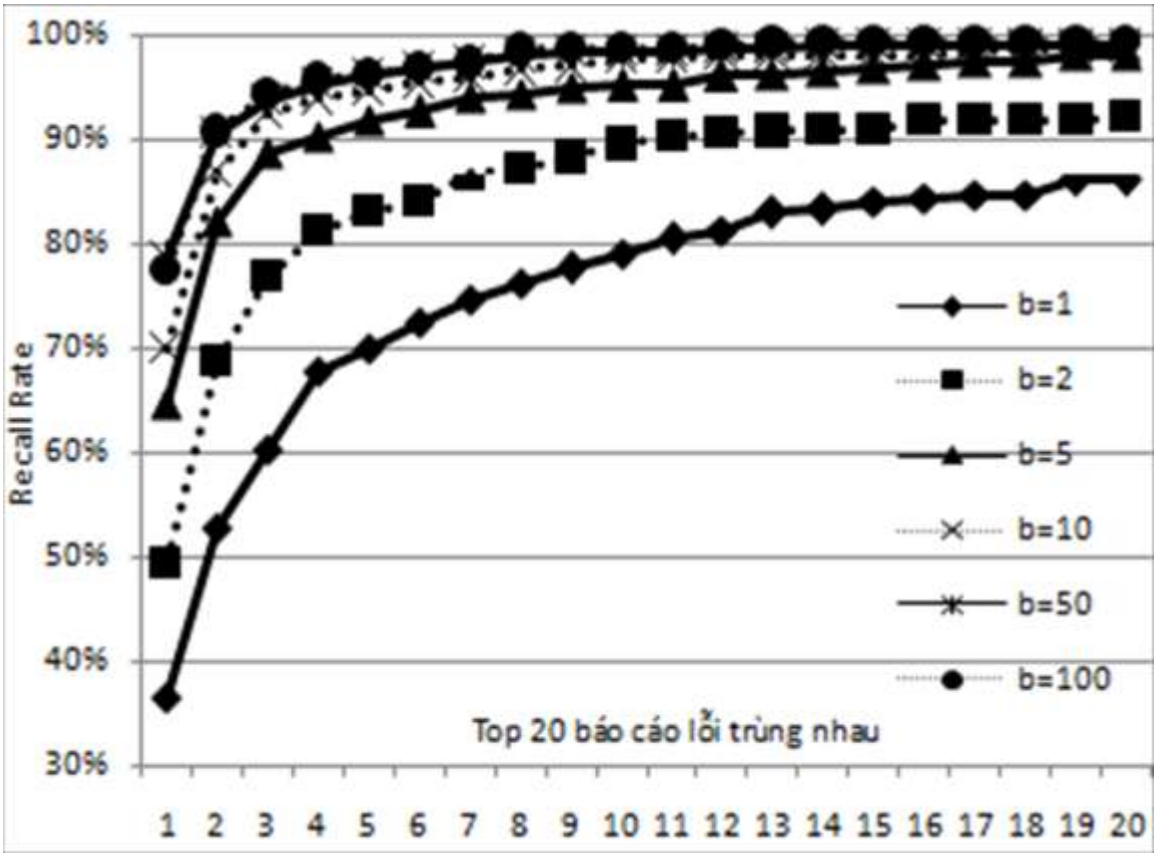
Hình 7: So sánh 4 loại đặc điểm trọng lượng từ

2. Tham số b

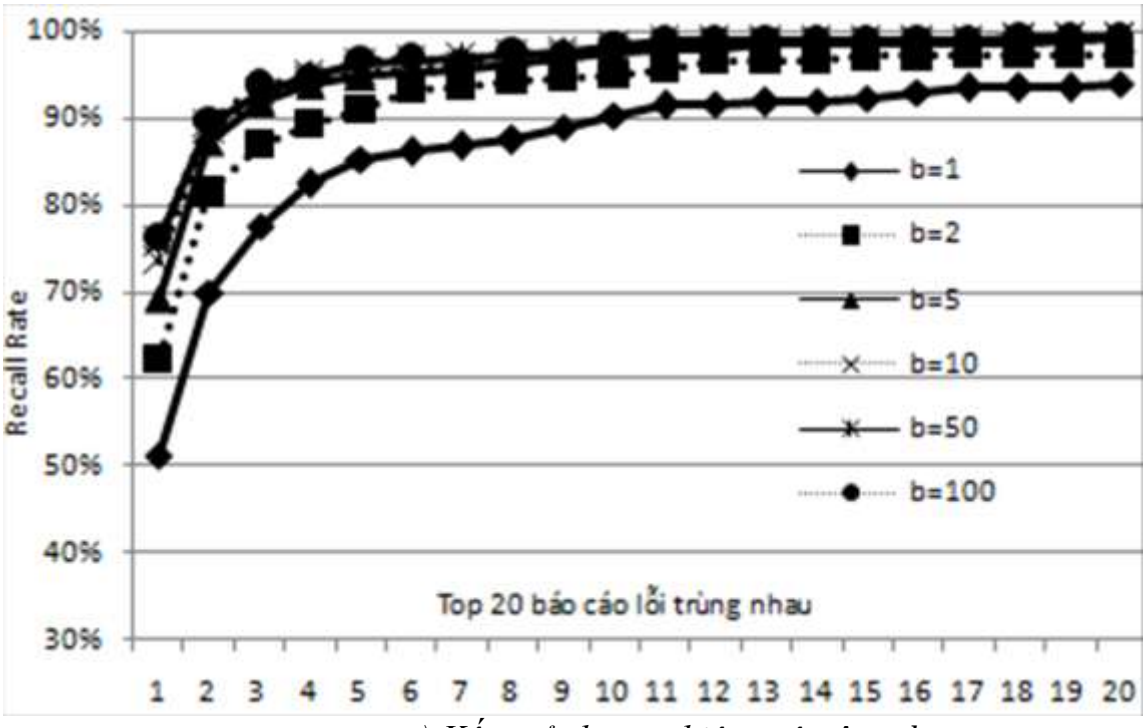
Tham số *b* đóng vai trò quan trọng trong EXP-TF-DF, do đó việc thực nghiệm để tìm ra giá trị tốt nhất cho tham số *b* là cần thiết để tìm ra giá trị *b* tốt nhất trong ECCI. Kết quả thực nghiệm cho thấy rằng giá trị *b* không có thay đổi nhiều khi *b* lớn hơn 50, trừ dự án SVN có tác động nhỏ nhưng không đáng kể. Do đó trong ECCI đã sử dụng *b*=50 cho các thực nghiệm còn lại. Tuy nhiên giá trị *b* có thể sẽ có thay đổi tùy dữ liệu thực nghiệm. Hình 8 cho thấy kết quả thực nghiệm đối với tham số *b*.



a) Kết quả thực nghiệm trên SVN



b) Kết quả thực nghiệm trên Argo UML

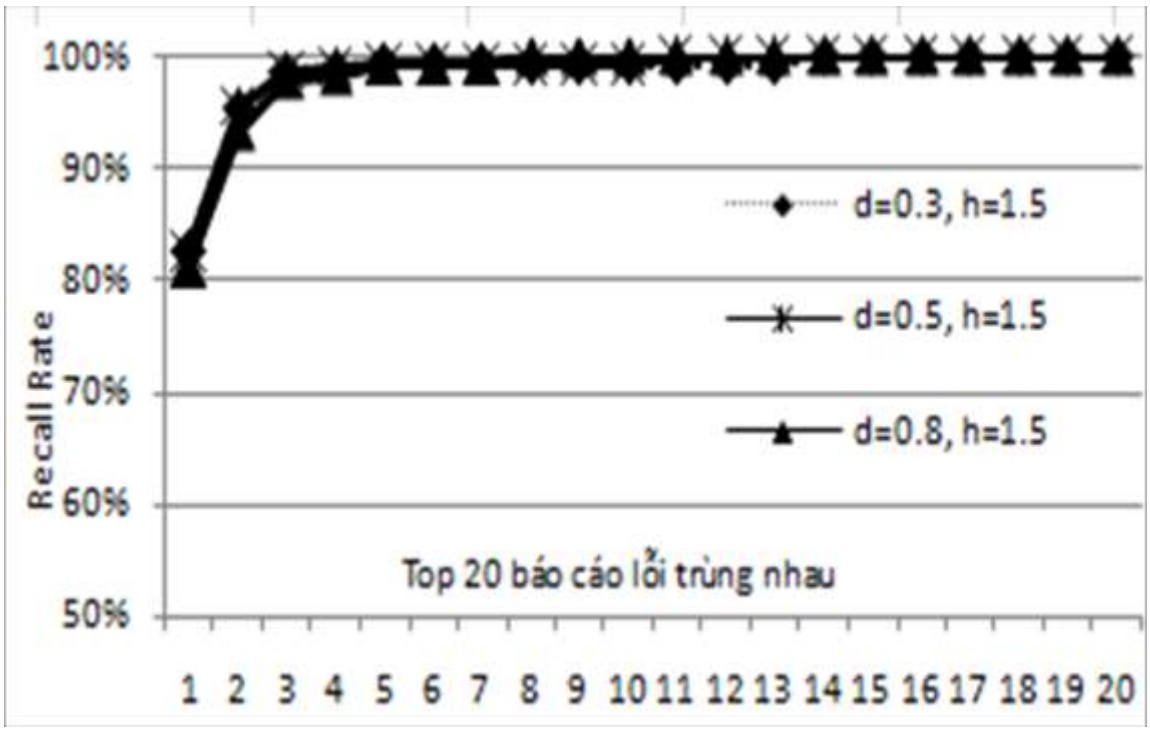


c) Kết quả thực nghiệm trên Apache

Hình 8: Tìm giá trị tốt nhất cho tham số b

3. Tham số d và h

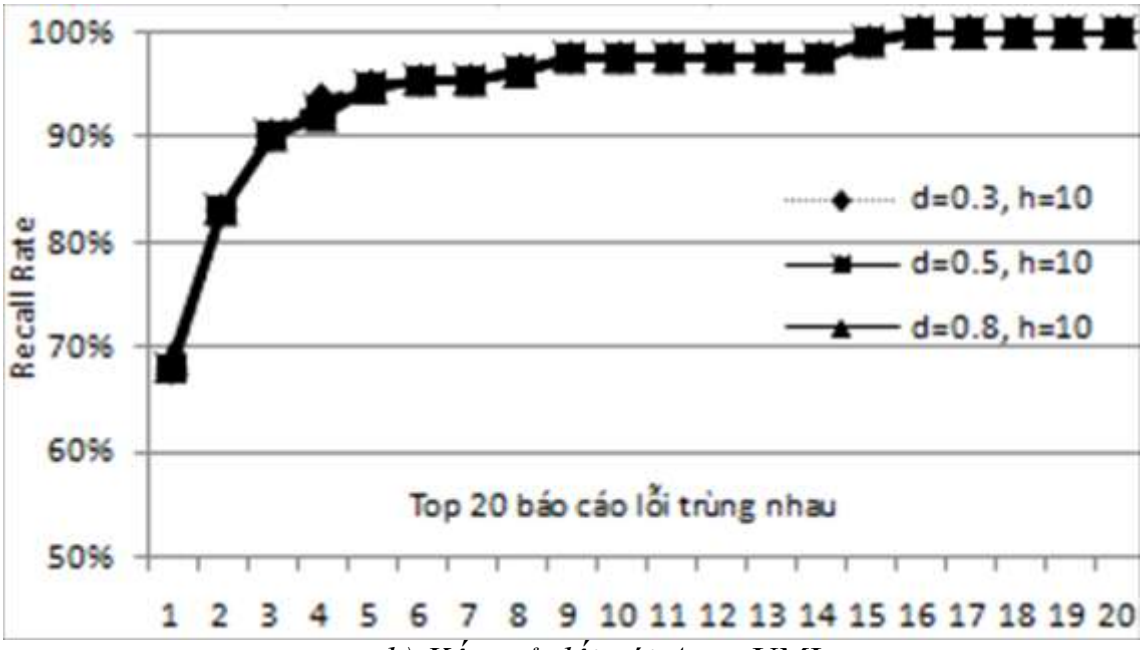
Tham số *d* và *h* cũng có ảnh hưởng trực tiếp đến EXP-TF-DF, do đó việc xác định giá trị tốt nhất cho *d* và *h* cũng góp phần quan trọng trong phương pháp



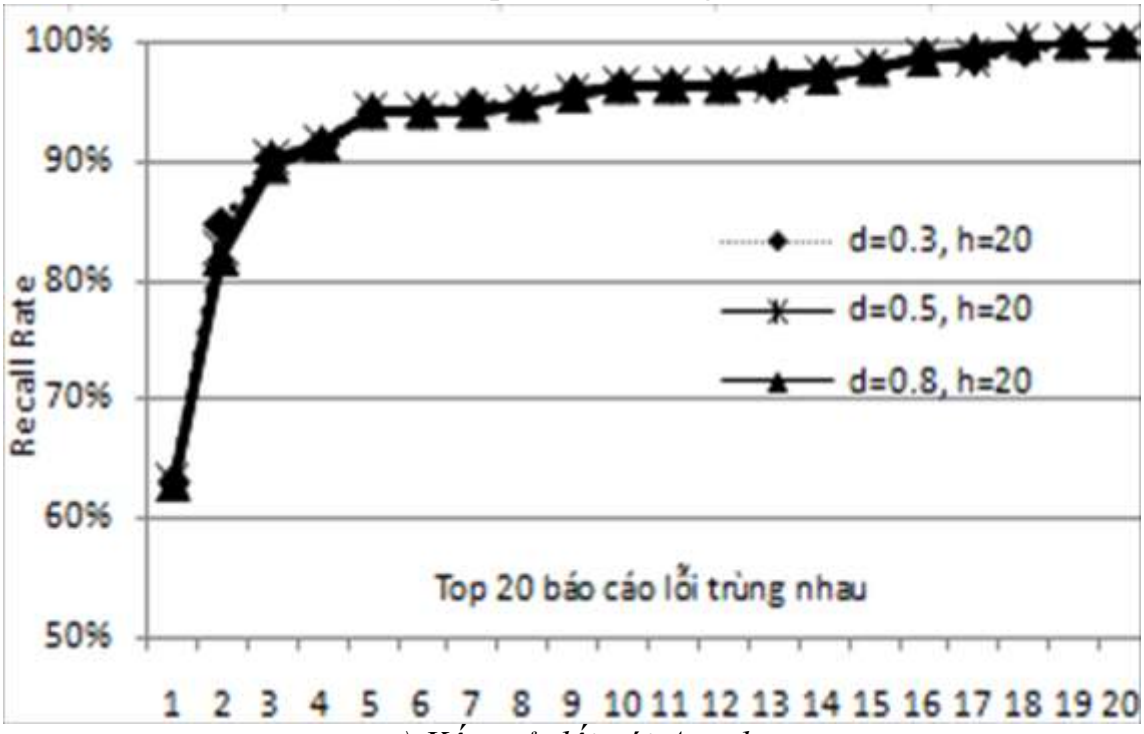
a) Kết quả đối với SVN

ECCI. Do đó việc thực nghiệm cũng được tiến hành để thấy sự ảnh hưởng của d và h trong EXP-TF-DF, với $0.3 \leq d \leq 0.8$ và $1.5 \leq h \leq 20$. Do có nhiều sự kết hợp giá trị giữa d và h, trong bài báo này chỉ trình bày một

vài trường hợp để minh họa chính để tìm ra giá trị tốt nhất cho d và h. Từ kết quả thực nghiệm như trong hình 9 đã xác định được giá trị tốt nhất cho d và h với d=0.3, h=1.5. Tuy nhiên giá trị này có thể thay đổi tùy theo những tập dữ liệu khác nhau.



b) Kết quả đối với Argo UML

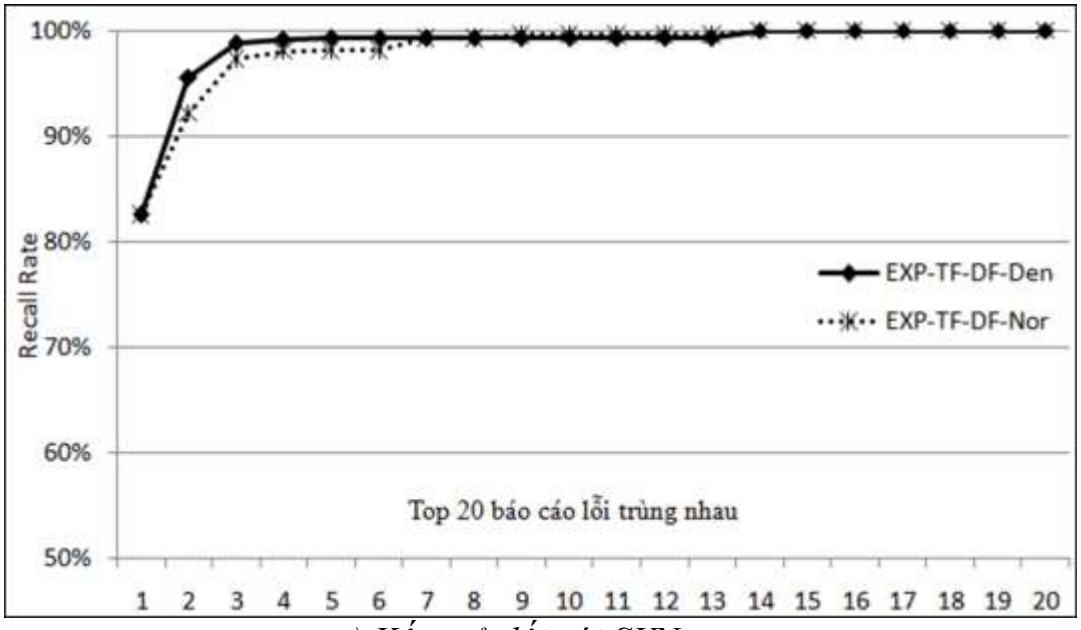


c) Kết quả đối với Apache

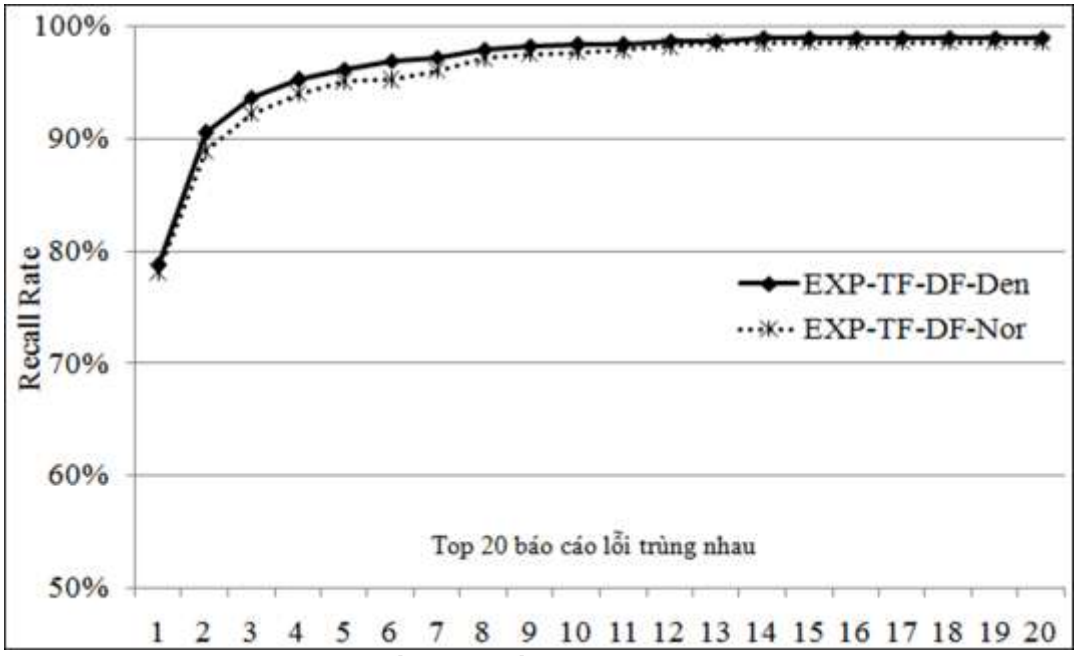
Hình 9: Tìm giá trị tốt nhất cho tham số d và h trên SVN

Denormalized cosine measure

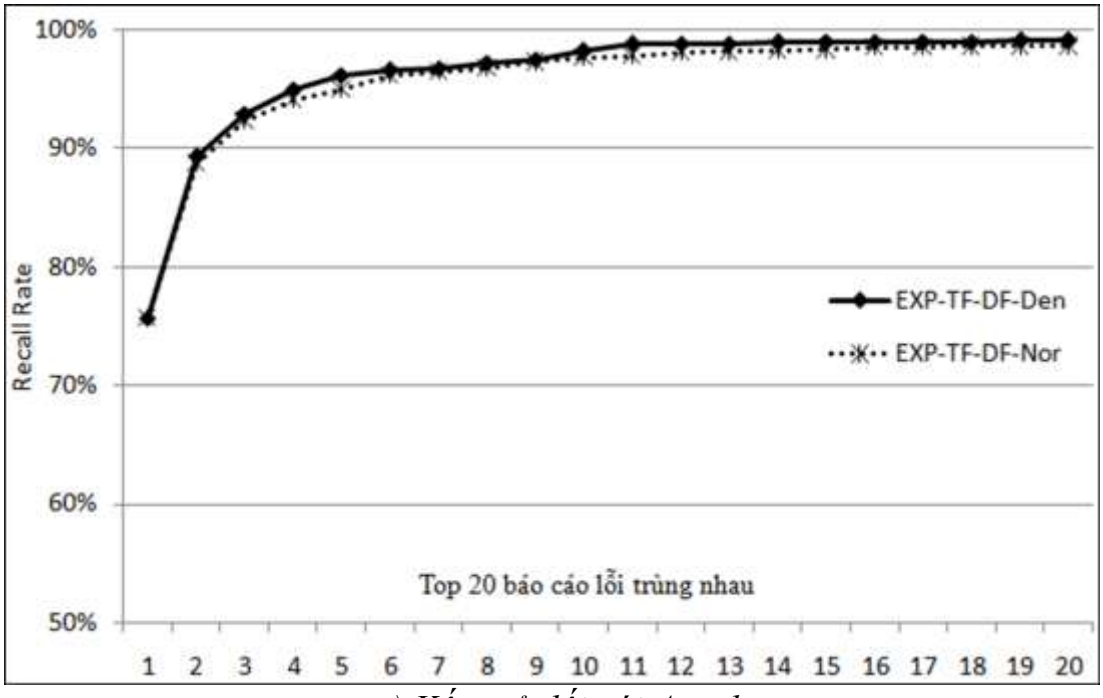
Trong CFC, việc tính độ giống nhau giữa các báo cáo lỗi sử dụng denormalized cosine measure. Để thấy rõ sự hiệu quả của nó so với cách truyền thống sử dụng nomalized cosine, việc thực nghiệm được tiến hành để so sánh hai phương pháp này. Quan sát kết quả thực nghiệm cho thấy rằng phương pháp denormalized cosine cho kết quả tốt hơn phương pháp normalized cosine trong cả ba dự án. Hình 10a đến 10c thể hiện kết quả so sánh giữa hai loại đo lường.



a) Kết quả đối với SVN



b) Kết quả đối với Argo UML

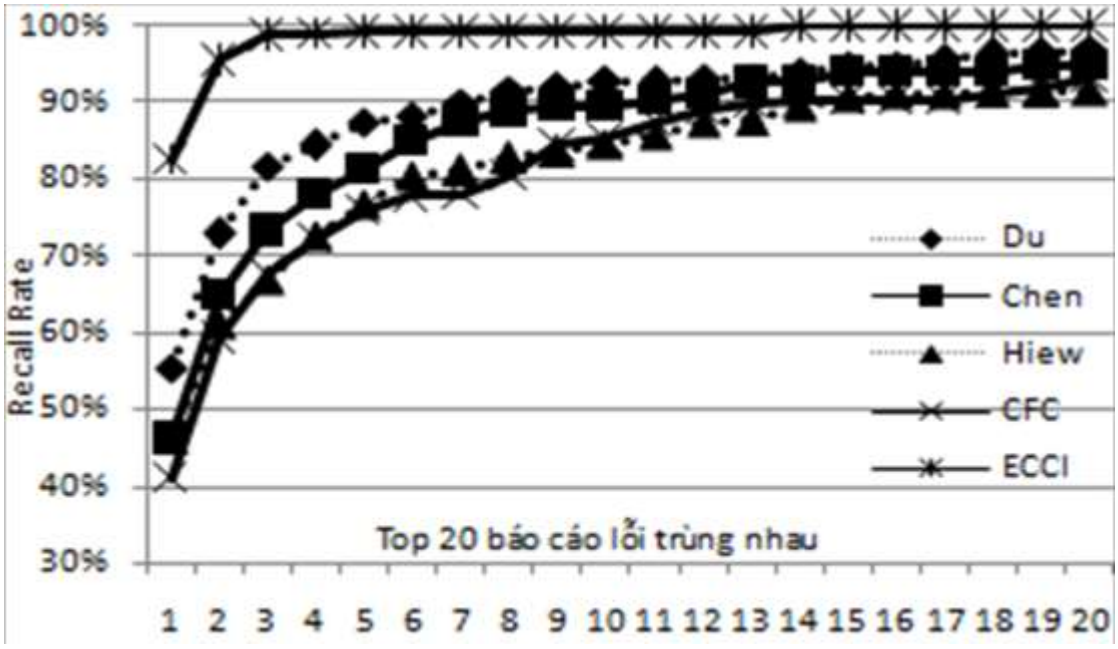


c) Kết quả đối với Apache

Hình 10: Kết quả so sánh hiệu quả giữa Denormalized và nomalized cosine

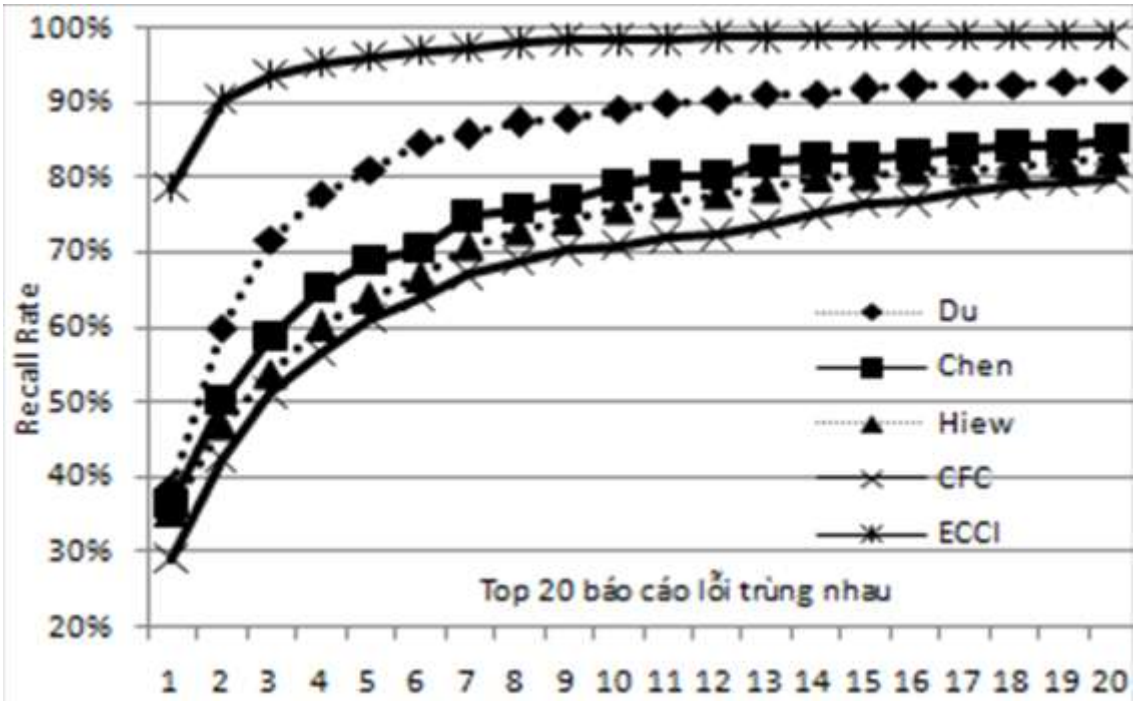
4. So sánh phương pháp ECCI với các phương pháp khác

Để thấy sự hiệu quả của ECCI với một số phương pháp dò tìm trùng nhau đã được công bố trước đây. Cụ thể là phương pháp của Hiew [2], Chen[3], Du [4], CFC[5], thực nghiệm đã tiến hành để so sánh, kết quả so sánh được thấy như hình 11a đến 11c. Từ kết quả thực nghiệm, chúng ta thấy rằng phương pháp ECCI cho kết quả dò tìm tốt hơn so với các phương pháp khác. Điều này

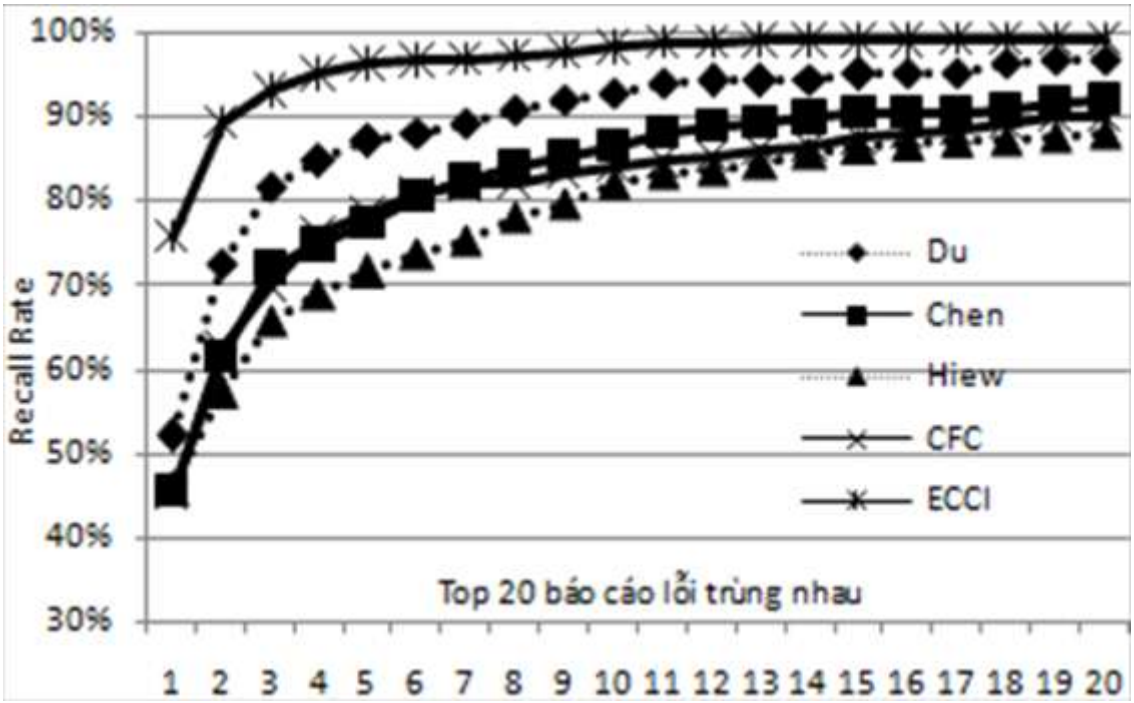


a) Kết quả đối với SVN

cho thấy sự hiệu quả của ECCI, khi xem xét các yếu tố liên quan đến thông tin lớp dựa vào centroid, trong việc tính trọng số của từ trong các báo cáo lỗi, cũng như hiệu quả của cách dùng phương pháp denormalized cosine.



b) Kết quả đối với Argo UML



c) Kết quả đối với SVN

Hình 11: So sách phương pháp ECCI với các phương pháp khác

PHẦN KẾT LUẬN

1. Kết quả đề tài và thảo luận

Việc dò tìm trùng nhau của những báo cáo lỗi là một trong những vấn đề quan trọng trong việc bảo trì phần mềm trong những năm gần đây. Trong bài báo này giới thiệu một phương pháp dò tìm dựa vào thông tin centroid lớp mở rộng (ECCI) để cải tiến việc thực thi dò tìm những báo cáo lỗi trùng nhau. Kết quả thực nghiệm từ ba dự án mã nguồn mở cho thấy phương pháp này mang lại hiệu quả cao trong việc dò tìm các báo cáo lỗi trùng nhau, đặc biệt là khi so sánh với các phương pháp được giới thiệu trước đây, phương pháp ECCI đã cho kết quả tốt hơn và hiệu quả hơn trong việc dò tìm những báo cáo lỗi trùng nhau khoảng 10% so với các phương pháp trước đó.

2. Kiến nghị

Đề tài nghiên cứu này là bước đầu để những nghiên cứu tiếp theo có thể ứng dụng nghiên cứu thử nghiệm đối với những văn bản tiếng việt, trong đó có liên quan đến việc kiểm tra đạo văn của sinh viên trong việc làm báo cáo đồ án, hay luận văn tốt nghiệp. Tuy nhiên vấn đề này đòi hỏi cần một nhóm nghiên cứu chuyên sâu với sự quyết tâm cao, ngoài ra phương pháp quản lý đồ án, báo cáo, luận văn của sinh viên cũng phải được quản lý online tập trung, khi đó mới có thể thực hiện được.

TÀI LIỆU THAM KHẢO

[1] Vincent Boisselle, Bram Adams MCIS, Polytechnique Montreal, Québec, "The Impact of Cross-Distribution Bug Duplicates, Empirical Study on Debian and Ubuntu", IEEE 15th International Working Conference on Source Code Analysis and Manipulation (SCAM),2015, page 131-140.

[2] Lyndon Hiew, "Assisted Detection of Duplicate Bug Reports," Master Thesis, The University of British Columbia, May2006.

[3] Zhi-Hao Chen, "Duplicate Detection on Bug Reports using N-Gram Features and Cluster Shrinkage", Master Thesis,Yuan Ze University, Jul.2011.

[4] Hung Hsueh Du, "A study of Duplication Detection Methods for Bug Reports based On BM25 Feature Weighting," MasterThesis,Yuan Ze University, Nov.2011.

[5] Hu Guan,JingyuZhou,andMinyiGuo,"AClass-Feature-Centroid Classifier for Text Categorization" in Proceedings of the18th International Conference on World Wide Web (WWW2009), 2009, pp.201–210.

[6] Eui-Hong Hanand George Karypis, "Centroid-Based Document Classification: Analysis and Experimental Results," in Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'00), 2000, pp.424–431.

[7] StephenE.Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford, "OkapiatTREC-3," in Proceedings of the Third Text Retrieval Conference(TREC-3),1994,pp.109–126.

[8] Xiao yan Zhang, TingWang, Xiao bo Liang, Feng Ao, and YanLi, "A Class-based Feature Weighting Method for Text Classification," Journal of Computational Information System, vol.3, pp.965–972, 2012.

[9] Xiao yin Wang, Lu Zhang, Tao Xie, John Anvik, and Jiasu Sun, "An Approach to DetectingDuplicate Bug Reports using Natural Language and Execution Information," in Proceedings of the 30th International Conference on Software Engineering (ICSE '08), 2008, pp. 461–470.

[10] Akihiro Tsuruda, Yuki Manabe, Masayoshi Aritsugi, "Can We Detect Bug Report Duplication with Unfinished Bug Reports?", Software Engineering Conference (APSEC) 2015 Asia-Pacific, *pp. 151-158, 2015, ISSN 1530-1362.*

[11] Chao-Yuan Lee, Dan-Dan Hu, Zhong-Yi Feng, Cheng-Zen Yang, "Mining Temporal Information to Improve Duplication Detection on Bug Reports", Advanced Applied Informatics (IIAI-AAI) 2015 IIAI 4th International Congress on, pp. 551-555, 2015.

[12] Chengnian Sun, David Lo, Xiaoyin Wang, Jing Jiang, and Siau-Cheng Khoo, “Discriminative model approach towards accurate duplicate bug report retrieval”. In ICSE 2010: Proceedings of the 32nd international conference on Software Engineering, Cape Town, South Africa, 2010. IEEE Computer Society.