TỰ ĐỘNG TÌM NHỮNG BÁO CÁO LỖI TRÙNG NHAU SỬ DỤNG KỸ THUẬT N-GRAM VÀ CLUSTER SHRINKAGE

Nhan Minh Phúc *

Tóm tắt

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 rất quan trọng và cần thiết, giúp tiết kiệm thời gian và công sức cho con người, trong những báo cáo mới được gởi đến. Bài báo giới thiệu một phương pháp mới sử dụng hai kỹ thuật: n-gram và cluster shrinkage. Phương pháp đã được thực nghiệm trên ba dự án phần mềm mã nguồn mở là Apache, ArgoUML, và SVN. Kết quả thực nghiệm chỉ ra rằng phương pháp được giới thiệu có hiệu quả cải tiến việc thực thi dò tìm khi được so sánh với những phương pháp trước đây.

Từ khóa: Báo cáo lỗi, dò tìm lỗi trùng nhau, đặc điểm N-gram, phân tích báo cáo lỗi, Cluster Shrinkage.

Abstract

For many open source projects, the number of reports about duplication occupies a significant percentage of the bug repositor. Therefore, automatic the identification of duplication error reports are very important and necessary and helps saving time and effort in searching for the duplicate bug reports out of any incoming ones. This paper presents a new approach using two techniques: n-gram and cluster shrinkage. Such approach has been experimented on three popular open source projects as Apache, Argo UML, and SVN. The experimental results show that the proposed method can effectively improve the detection performance as compared with the previous methods.

Keywords: Bug Reports, Duplicate Bug Detection, N-gram feature, Bug Report Analysis, Cluster Shrinkage.

1. Giới thiệu

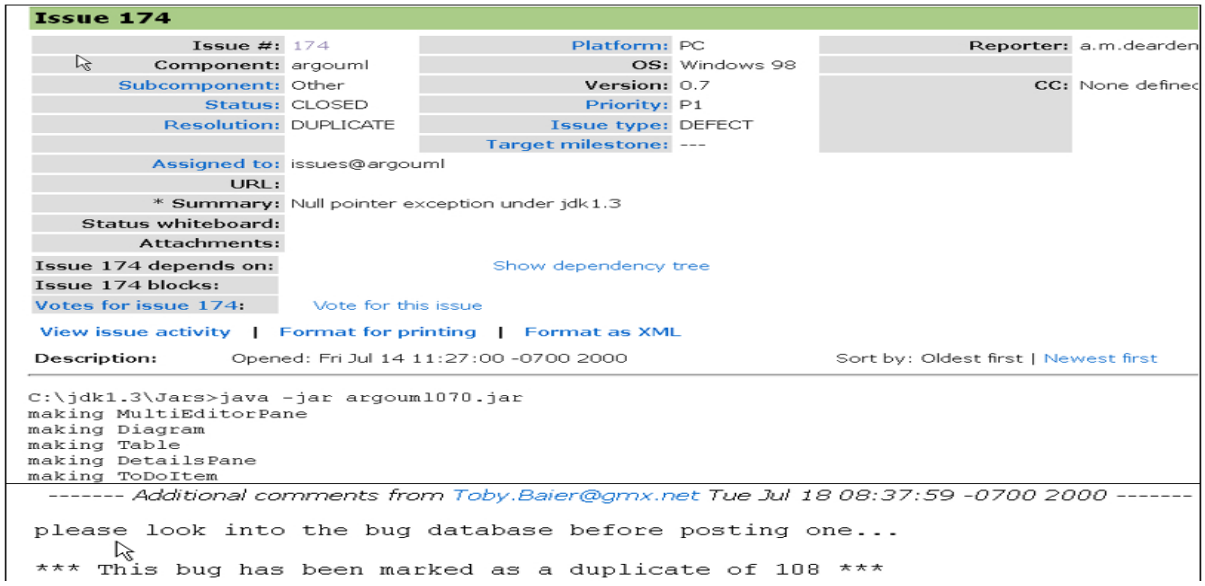
Trong vấn đề bảo trì phần mềm, việc tìm ra những lỗi cũng như những vấn đề không bình thường là một xử lý quan trọng để tránh những rủi ro. Thông thường, những tình huống này sẽ được miêu tả lại và gởi đến hệ thống quản lý báo cáo lỗi như Bugzilla, Eclipse… Sau khi những báo cáo lỗi được gởi, một hoặc nhiều người sẽ được giao nhiệm vụ phân tích những lỗi này và chuyển đến những lập trình viên phù hợp cho việc xử lý lỗi. Theo những bài báo gần đây, vấn đề dò tìm lỗi trùng nhau đang nhận được nhiều sự quan tâm của các nhà nghiên cứu, lý do chính là do số lượng báo cáo lỗi trùng nhau đã tăng đến 36%. Cụ thể dự án của Eclipse được thống kê từ tháng 10/2001 đến tháng 8/2005 có 18.165 báo cáo lỗi, trong đó, những lỗi trùng nhau chiếm tới 20%. Ngoài ra, dữ liệu của Firefox được thống kê từ tháng 5/2003 đến tháng 8/2005 có 2.013 báo cáo lỗi được gởi, trong đó, 30% là những báo cáo lỗi trùng nhau. Số liệu thống kê cho thấy số lượng những báo cáo lỗi trùng nhau là rất lớn, điều này cho thấy tầm quan trọng của việc đưa ra những giải pháp trong việc xử lý lỗi trùng nhau là hết sức cần thiết và cấp bách. Vì vậy, việc nhận biết những báo cáo lỗi 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, lý do là vì họ có thể cung cấp nhiều thông tin hơn so với những báo cáo lỗi được gởi trước đó.

2. Vấn đề dò tìm lỗi trùng nhau

Vấn đề lỗi trùng nhau có thể được phân loại bằng việc xác định hai hoặc nhiều hơn những báo cáo lỗi mà nó mô tả có cùng lỗi phần mềm. Theo những bài báo trước đây, những báo cáo lỗi trùng nhau có thể được chia thành hai loại. Loại thứ nhất mô tả những báo cáo lỗi xảy ra trong cùng tình huống. Loại thứ hai miêu tả những báo cáo lỗi khác nhau với cùng nguồn gốc của lỗi phần mềm. Do những báo cáo lỗi loại thứ hai thường được mô

* *Thạc sĩ - Khoa Kỹ thuật & Công nghệ, Đại học Trà Vinh*

tả bởi những từ vựng khác nhau cho những báo cáo lỗi khác nhau, vì vậy việc dò tìm lỗi trùng nhau có thể không hiệu quả bởi việc những báo cáo này chỉ được mô tả bởi những thông tin văn bản. Để dò tìm hiệu quả loại thứ hai, nó đòi hỏi những thông tin báo cáo lỗi cụ thể hơn như theo dõi việc thực thi chương trình. Tuy nhiên, vấn đề này lại liên quan đến những thông tin cá nhân, khi đó, nghiên cứu này chỉ tập trung vào phương pháp dò tìm theo loại thứ nhất, nghĩa là chúng ta chỉ xem xét những báo cáo lỗi với những mô tả lỗi bằng thông tin văn bản.



Hình 1.1. Một ví dụ về báo cáo lỗi

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,… Ngoài ra, nó cũng có phần bình luận cho những người báo cáo lỗi khác bình luận. Hình 1.1 là một ví dụ của dự án Argo UML, trong đó, một báo cáo lỗi được gởi đến hệ thống theo dõi lỗi bởi người dùng. Trong trường mô tả, A.m.dearden đã cung cấp thông tin lỗi ngoại lệ khi thi hành trong Argo UML. 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. Hình 1.1 cho thấy lỗi báo cáo này có mã số 174 và được xác định là lỗi trùng nhau với lỗi báo cáo mã số 108. Trong trường hợp này, hai báo cáo lỗi được gọi là trùng nhau và có cùng nhóm lỗi.

Vấn đề trùng nhau trong nghiên cứu này được xử lý như sau. Đối với một dự án phần mềm, những báo cáo lỗi đã tồn tại trước đây sẽ được xử lý đầu tiên bằng cách phân loại thành n nhóm báo cáo. Mỗi nhóm báo cáo sẽ có một báo cáo lỗi chính. Nếu một nhóm báo cáo có nhiều hơn một cáo cáo lỗi, khi đó những báo cáo lỗi trong nhóm báo cáo sẽ có mối quan hệ trùng nhau. Khi có một báo cáo lỗi được gởi đến, việc dò tìm trùng nhau được thực hiện để đưa ra một danh sách những báo cáo lỗi gần giống nhất với những báo cáo lỗi trong nhóm báo cáo. Mối quan hệ trùng nhau được xác định nếu thỏa một trong các điều kiện sau:

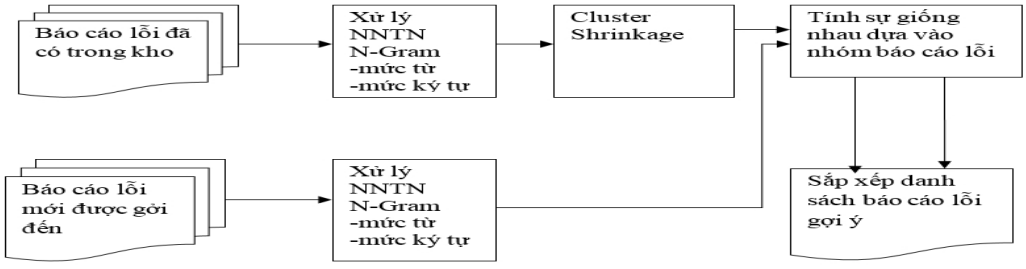
1. Cho một báo cáo lỗi chính BR_m, một báo cáo lỗi BR_i sẽ được đánh dấu là trùng nhau với BR_m trong hệ thống theo dõi lỗi và trạng thái báo cáo lỗi khi đó là đóng (Closed).
2. Cho hai báo cáo lỗi BR_i và BR_j, nếu chúng được đánh dấu như trùng nhau của báo cáo BR_m, BR_i sẽ được xem là trùng nhau của BR_j và ngược lại.
3. Nếu có một báo cáo lỗi BR_k khác mà được đánh dấu như trùng nhau của BR_i, thì BR_k cũng là trùng nhau của BR_m. Trường hợp này được gọi là bắc cầu.

3. Phương pháp dò tìm lỗi trùng nhau

Phương pháp được giới thiệu sử dụng kỹ thuật xử lý ngôn ngữ tự nhiên (Natural Language Proccessing), N-gram, và Cluster Shrinkage. Cho một dự án phần mềm, những báo cáo lỗi đã được báo cáo trước đây được phân loại sang n nhóm báo cáo. Nhóm báo cáo lỗi sẽ được xây dựng dựa vào

phần bình luận của báo cáo lỗi để tạo ra một tập tin gọi là Mapping File. Trong một nhóm mà nó có nhiều hơn một báo cáo lỗi, báo cáo lỗi sau cùng sẽ dùng làm dữ liệu kiểm tra (test data). Nói cách khác, mã lỗi lớn nhất trong mỗi nhóm là báo cáo lỗi mới nhất (báo cáo được nhận sau cùng). Những báo cáo lỗi khác đã tồn tại trước đó trong nhóm được gọi là báo cáo lỗi đã tồn tại. Trong quá trình phân tích và quan sát những báo cáo lỗi, chúng tôi phát hiện ra rằng những báo cáo lỗi có sự liên quan về mặt ngữ nghĩa. Lý do chính là người gởi báo cáo lỗi có thể không mô tả một cách chi tiết lỗi mà chỉ sử dụng những từ ngữ khác nhau để mô tả cùng một loại lỗi. Ngoài ra, họ cũng sử dụng nhiều từ ghép khi viết báo cáo lỗi. Khi đó, kỹ thuật n-gram được sử dụng để trích ra những thông tin từ những khác biệt này. Kỹ thuật này có thể cải thiện việc xác định sự giống nhau giữa những báo cáo lỗi có cùng nhóm báo cáo. Khi đó, chúng tôi sử dụng kỹ thuật cluster shrinkage tiếp tục cải thiện việc nhận biết sự giống nhau bằng cách tính lại từ những đặc điểm trọng lượng của những báo cáo lỗi. Phương pháp được giới thiệu bao gồm bốn bước như sau:

1. Trích ra những đặc điểm cần thiết từ báo cáo lỗi.
2. Tính trọng lượng đặc điểm của mỗi từ trong báo cáo lỗi.
3. Xác định có hay không sự giống nhau giữa các báo cáo lỗi.
4. Tạo ra danh sách Top n các báo cáo lỗi trùng nhau.



Hình 3.1. Mô hình xử lý

3.1 Trích ra những đặc điểm cần thiết từ báo cáo lỗi

3.1.1 Những loại dữ liệu

Chúng tôi có hai loại báo cáo lỗi. Loại đầu được gọi là những báo cáo lỗi đã tồn tại, hay còn gọi là loại có sẵn. Loại thứ hai là những báo cáo lỗi mới được gởi đến là loại báo cáo lỗi có mã báo cáo lỗi lớn nhất trong tất cả nhóm báo cáo. Nói cách khác, lỗi này là lỗi sau cùng mà nó được gởi đến trong mỗi nhóm báo cáo lỗi. Báo cáo lỗi loại một có sẵn trong kho chứa lỗi của dự án phần mềm. Cho mỗi báo cáo lỗi vừa được gởi đến, nó sẽ có mã số lỗi lớn hơn những báo cáo lỗi đã tồn tại trước. Chúng tôi sẽ thiết kế những kỹ thuật cho những báo cáo lỗi đã tồn tại để giúp chúng ta tìm những báo cáo lỗi trùng nhau.

3.1.2 Nhóm báo cáo lỗi

Dựa vào thông tin được trích từ những báo cáo lỗi đã tồn tại, chúng tôi xây dựng một tập tin ánh xạ hay còn gọi là mapping file chứa nhóm báo cáo lỗi. Một ví dụ về tập tin ánh xạ này được minh họa trong Bảng 3.1. Trong đó, cột đầu tiên cho biết số nhóm báo cáo lỗi, cột hai hiển thị báo cáo lỗi vừa được gởi đến trong cùng nhóm, cột cuối cùng cho biết những báo cáo lỗi bị trùng nhau. Chú ý rằng, kích thước nhỏ nhất của nhóm báo cáo là 2 hay nói cách khác nhóm báo cáo nhỏ nhất là sự kết hợp chỉ với một báo cáo lỗi vừa được gởi đến và một báo cáo đã tồn tại trước đó và hai báo cáo lỗi này trùng nhau. Trong bảng 3.2, chúng ta có thể thấy hầu hết kích thước của nhóm báo cáo nằm trong khoảng từ 2 đến 4.

Bảng 3.1. Một ví dụ về file ánh xạ trong báo cáo lỗi trùng nhau trong phần mềm SVN

Số nhóm báo cáo lỗi	báo cáo lỗi mới	Những báo cáo lỗi trùng nhau
1	330	329
2	433	387
...	...	...
n	2610	2520, 2407, 525
...	...	...

3.2 Việc trích đặc điểm n-gram

Chúng tôi sử dụng mô hình không gian vector để xử lý những báo cáo lỗi. Trong bước này, chúng tôi sử dụng việc xử lý ngôn ngữ tự nhiên và kỹ thuật n-gram để giúp chúng tôi xây dựng vector báo cáo lỗi. Chúng tôi sử dụng Word Vector Tool (WVT), một công cụ hỗ trợ thư viện Java, để giúp chúng tôi tính các vector. Trong công cụ WVT, chúng tôi đã xây dựng một báo cáo lỗi bao gồm 3 phần. Phần một, chúng tôi sử dụng NLP để loại bỏ những ký tự không cần thiết trong báo cáo lỗi. Thứ hai chúng tôi sử dụng kỹ thuật n-gram ở mức ký tự để tìm sự giống nhau giữa những từ, cũng như tìm ra những từ gốc của những từ đã được viết tắt. Ngoài ra, nó cũng giúp tìm ra những từ ghép trong báo cáo lỗi. Bảng 3.3 là một ví dụ của một báo cáo lỗi có mã số lỗi là 330 và Bảng 3.4 minh họa một vector sau khi chúng tôi đã tiến hành tiền xử lý với NLP.

Bảng 3.2. Kích thước nhóm báo cáo lỗi trong các kho chứa lỗi

Kích thước	2	3	4	5	6	7	8	9	10	...	Tổng
SVN	103	24	6	1	1	0	0	0	0	...	135
ArgoUML	230	47	16	6	3	4	0	0	0	...	307
Apache	138	43	11	8	7	1	2	0	0	...	214
Eclipse	12390	3904	1279	548	278	156	79	48	45	...	18843
...					...	...					

Bảng 3.3. Một báo cáo lỗi trong SVN

SVN Bug Report 330
short description : 'svn ci' can't deal with mixed-revision working copies
Long Description : WHO : sussman — BUG_WHEN : 2001-03-19 10:41:03 — Ben rewrote the fs commit editor so that replace *() would call svn_fs_copy() ifthe base_rev argument was unexpected. This should result in a transaction beingbuilt that propely mirrors the mixed-revision working copy.However- we're now getting a conflict in merge() when we commit the transaction. Mike Pilato is looking into it.WHO : sussman — BUG_WHEN : 2001-03-19 10:48:59 — *** This bug has been marked as a duplicate of 329 ***|

Bảng 3.4. Một ví dụ về xử lý lỗi

	SVN Bug Report 330 Vector
NLP	result deal edit ...
Char-based *n*-gram	mixed-ixed-r xed-re ...
Word-based *n*-gram	mirrorsthemixed-revision mixedrevisionworking revisionworkingcopy ...

3.3 Tính trọng lượng đặc điểm

Chúng tôi sử dụng kỹ thuật cluster shrinkage để giúp tìm ra ngữ nghĩa của những báo cáo lỗi trùng nhau. Đầu tiên chúng tôi sẽ xác định trọng tâm (centroid) của nhóm báo cáo. Kế đến tất cả báo cáo lỗi sẽ được co (cluster shrinkage) về trọng tâm của nhóm.

1. Trọng tâm của nhóm: mỗi nhóm báo cáo lỗi có một trọng tâm (centroid) mà nó chứa tất cả thông tin trong nhóm của nó. Để tính trọng tâm của một nhóm báo cáo lỗi, chúng tôi sẽ tính vector trung bình của nhóm đó. Ưu điểm của phương pháp này là do những người gởi báo cáo lỗi không phải lúc nào cũng mô tả một cách chi tiết lỗi xảy ra, điều này làm cho việc tính sự giống nhau giữa các báo cáo ít hiệu quả, và hai báo cáo trùng nhau rất ít khi dùng cùng một số từ ít sử dụng, điều này gây khó khăn cho việc xác định những báo cáo lỗi trùng nhau. Vì vậy, centroid là một trong những giải pháp khắc phục trường hợp này.
2. Việc sử dụng cluster shrinkage: Sau khi tìm được centroid của nhóm báo cáo lỗi, chúng tôi sử dụng kỹ thuật cluster shrinkage để co tất cả báo cáo lỗi đến centroid của nhóm. Thuật toán được thể hiện như sau :

Cho mỗi nhóm báo cáo lỗi S

{

N là số báo cáo lỗi trong S:

Tính trọng tâm của nhóm báo cáo lỗi:

$C=\frac{1}{N}\sum_{\square}^{N} v$ với v là một vector báo cáo lỗi

Cho mỗi báo cáo lỗi $v \in S$ {

$v'=(1-\lambda)v+\lambda C$, v' là một vector báo lỗi mới được thay thế bởi v

trong đó $0 \leq \lambda \leq 1$

}

}

3.4. Tính sự giống nhau giữa các báo cáo lỗi

Chúng tôi cũng sử dụng cùng phương pháp tính cosine như những nghiên cứu trước đây.

$$Cos(\vec{X_i}, \vec{Y_j}) = \frac{\vec{X_i} \cdot \vec{Y_j}}{|\vec{X_i}| \cdot |\vec{Y_j}|}$$

Trong đó:

$\vec{X_i}$ biểu diễn cho vector đặc điểm của báo cáo lỗi mới gởi đến.

$\vec{Y_j}$ biểu diễn vector đặc điểm của mỗi báo cáo lỗi đã tồn tại trong dataset.

Phương pháp này có thể giúp việc tính toán sự giống nhau giữa hai báo cáo lỗi tốt hơn. Phương pháp tính sự giống nhau trong các báo cáo lỗi sử dụng trong bài báo này gọi là sự sắp xếp dựa vào nhóm báo cáo lỗi, có nghĩa là giá trị cosine sẽ được tính lại lần nữa trước khi xác định xem hai báo cáo lỗi có trùng nhau không. Tiếp theo, chúng ta tính trung bình cosine của những báo cáo lỗi trong nhóm báo cáo và so sánh báo cáo lỗi mới vừa gởi đến với tất cả báo cáo lỗi đã tồn tại với giá trị cosine mới và sắp xếp lại theo giá trị giống nhau giữa các báo cáo lỗi để xác định trùng nhau. Cách này có thể giải quyết phần nào những báo cáo lỗi trùng nhau mà có giá trị cosine thấp.

3.5 Danh sách Top n báo cáo tương tự

Việc sử dụng danh sách Top n báo cáo lỗi tương tự nhau có thể giúp người dùng tìm được những báo cáo lỗi trùng nhau. Chúng tôi sắp xếp danh sách từ 1 đến 22 báo cáo lỗi gần giống nhất với báo cáo lỗi vừa được gởi đến và quan sát kết quả thực nghiệm. Khi đó chúng tôi tiến hành so sánh với những phương pháp nghiên cứu trước đây, kết quả thấy rằng, phương pháp của chúng tôi đã thực hiện tốt hơn những phương pháp đã được giới thiệu trước đó.

4 .Thực nghiệm

4.1 Môi trường thực nghiệm

Chúng tôi đã tiến hành thực nghiệm với ba kho báo cáo lỗi của những dự án phần mềm mở là Argo UML, Apache , và SVN. Thống kê chi tiết về ba kho phần mềm này được mô tả trong Bảng 4.1.

Bảng 4.1. Thông tin về datasets của ba 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

4.2 Môi trường cài đặt

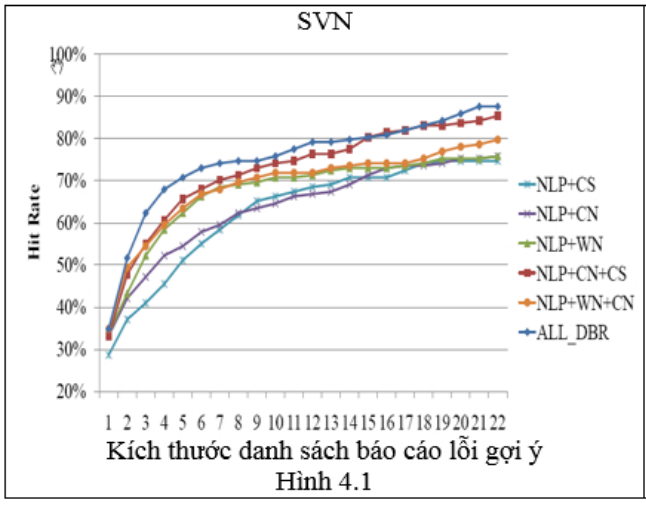
Phương pháp được giới thiệu sử dụng ba tham số, tham số đầu tiên nc chứa kích thước n-gram ký tự. Tham số thứ hai nw là chiều dài n-gram tính bằng từ. Cả hai tham số này đều có ảnh hưởng trực tiếp đến việc trích đặc điểm trong báo cáo lỗi. Tham số cuối cùng là CS. Trong các thực nghiệm với phương pháp này, nc=6, nw=3 và CS=0.9 cho kết quả thực thi tốt nhất.

Để đánh giá phương pháp dò tìm, chúng tôi sử dụng đơn vị đo lường gọi là Hit rate mà nó được tính dựa trên bao nhiêu báo cáo lỗi có thể được dò tìm đúng trong danh sách những báo cáo lỗi trùng nhau và nó được định nghĩa như sau:

$$\text{Hit rate} = \frac{\text{Số những dự đoán đúng}}{\text{Tổng số những báo cáo lỗi trùng nhau}}$$

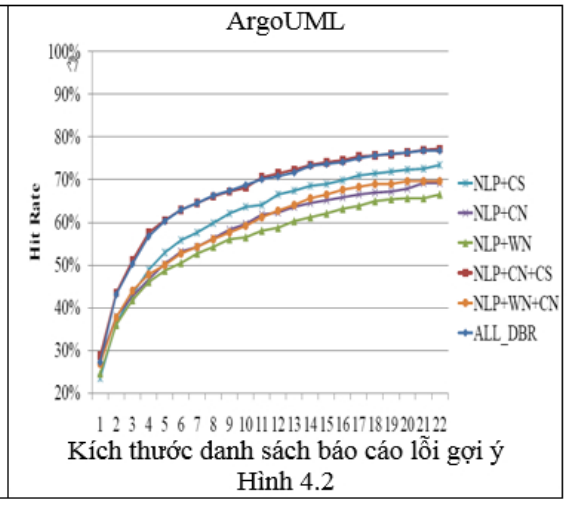
4.3 Nghiên cứu kết quả thực nghiệm

Việc đánh giá kết quả thực nghiệm dựa vào danh sách báo cáo lỗi trùng nhau gợi ý. Danh sách này giúp cho việc phân tích để tìm ra những báo cáo trùng nhau một cách nhanh chóng. Trong bài báo chúng tôi trình bày hai kết quả thực nghiệm. Thực nghiệm thứ nhất nghiên cứu sự kết hợp của những kỹ thuật khác nhau. Từ Hình 4.1 đến 4.3, quy định NLP nghĩa là kỹ thuật xử lý ngôn ngữ tự nhiên đơn giản, CN nghĩa n-gram mức ký tự và WN nghĩa là n-gram ở mức từ, CS là cluster shrinkage và ALL nghĩa là kết hợp của tất cả kỹ thuật trên. Từ những hình này, chúng ta thấy rằng ALL cho kết quả tốt nhất. Thực nghiệm cũng được thực hiện với các tham số khác nhau của CS, và kết quả cho thấy rằng CS=0.9 và CS=1.0 cho kết quả tốt nhất. Tuy nhiên, do CS=0.9 thì tốt hơn CS=1.0 một chút trong cả ba dự án nên trong những phần thực nghiệm còn lại đều sử dụng CS=0.9. Ngoài ra, tham số nc cũng được thực nghiệm để tìm ra giá trị tốt tốt nhất. Kết quả cho thấy rằng không có nhiều sự khác biệt giữa những giá trị này. Trong những thực nghiệm trên, sử dụng nc=6, nw=3 là do nó cho kết quả tốt nhất so với những giá trị khác mặc dù, sự khác biệt không nhiều. Lý do chính là chiều dài một từ trung bình trong báo cáo lỗi nằm trong khoảng từ 5 đến 6 ký tự. Hình 4.4 đến Hình 4.6 chúng tôi so sánh phương pháp của chúng tôi với những phương pháp đã được giới thiệu trước đây. Cụ thể là với phương pháp của Hiew, Runeson et al và Sureke et al. Kết quả cho thấy rằng phương pháp của chúng tôi cho kết quả tốt hơn những phương pháp của họ ít nhất 5%.



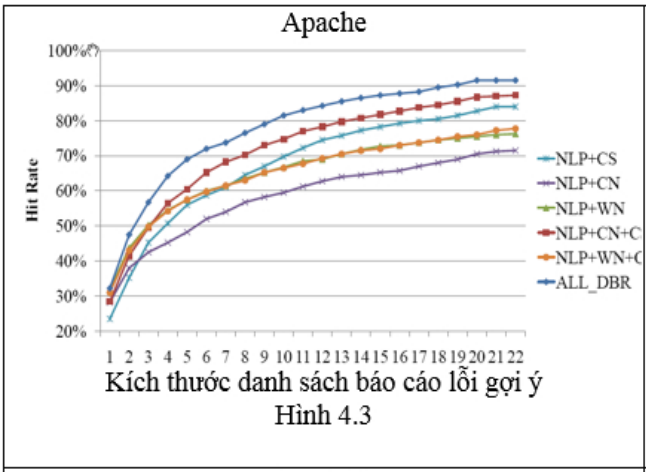
Kích thước danh sách báo cáo lỗi gợi ý

Hình 4.1



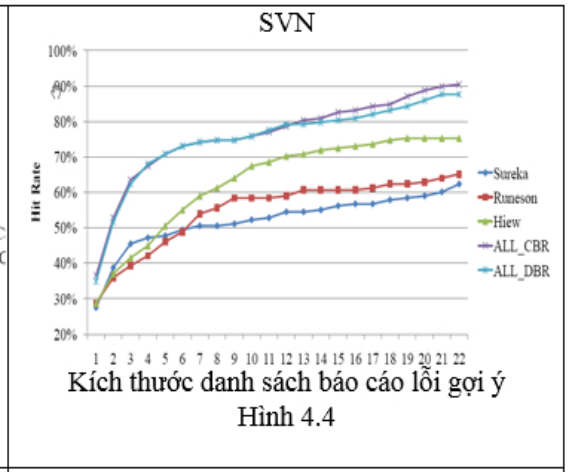
Kích thước danh sách báo cáo lỗi gợi ý

Hình 4.2



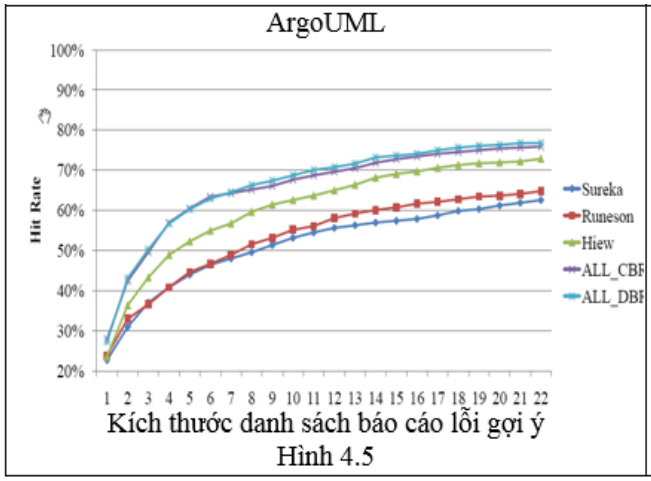
Kích thước danh sách báo cáo lỗi gợi ý

Hình 4.3

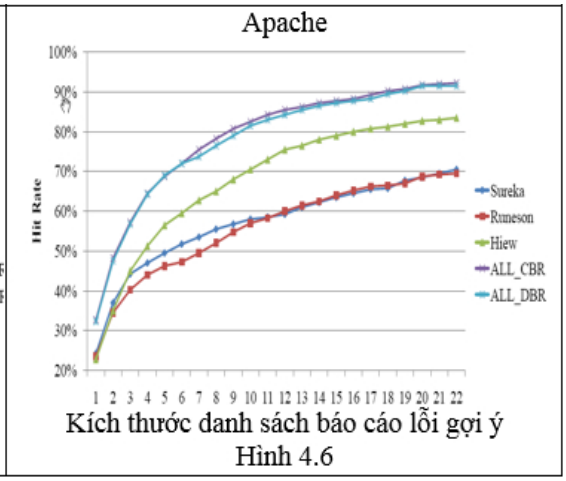


Kích thước danh sách báo cáo lỗi gợi ý

Hình 4.4



Hình 4.5



Hình 4.6

5. Kết luận

Việc dò tìm báo cáo lỗi trùng nhau là một vấn đề quan trọng trong việc bảo trì phần mềm trong những năm gần đây. Trong bài báo này, chúng tôi muốn giới thiệu một phương pháp mới sử dụng kỹ thuật n-gram và cluster shrinkage. Kết quả thực nghiệm từ ba dự án mã nguồn mở cho thấy phương pháp của chúng tôi 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.

Tài liệu tham khảo

Ashish Sureka and Pankaj Jalote. 2010. *Detecting Duplicate Bug Report Using Character N-Gram-based Features*. The 17th Asia Pacific Software Engineering Conference. pp. 366–374.

John Anvik, Lyndon Hiew, and Gail C. Murphy. 2005. *Coping with an Open Bug Repository*. Proceedings of the 2005 OOPSLA workshop on Eclipse technology eX-change (eclipse ’05). pp. 35–39.

John Anvik, Lyndon Hiew, and Gail C. Murphy. 2006. *Who Should Fix this Bug?* Proceedings of the 28th International Conference on Software Engineerin (ICSE’06). New York, NY, USA: ACM. pp. 361–370.

Lyndon Hiew. May 2006. *Assisted Detection of Duplicate Bug Reports*. Master Thesis, The University of British Columbia.

Per Runeson, Magnus Alexandersson, and Oskar Nyholm. 2007. *Detection of Duplicate Defect Reports Using Natural Language Processing*. The 29th International Conference on Software Engineering (ICSE 2007). pp. 499–510.

Xiaoyin Wang, Lu Zhang, Tao Xie, John Anvik, and Jiasu Sun.2008. *An Approach to Detecting Duplicate Bug Reports using Natural Language and Execution Information*. The 30th International Conference on Software Engineering (ICSE ’08).New York, NY, USA: ACM. pp. 461–47.

Yguaratã Cerqueira Cavalcanti, Eduardo Santana de Almeida, Carlos Eduardo Al- buquerque da Cunha, Daniel Lucre´dio, and Silvio Romero de Lemos Meira. 2010. *An Initial Study on the Bug Report Duplication Problem*. The 14th European Conference on Software Maintenance and Reengineering. pp. 264–276.