Information Retrieval Based Nearest Neighbor Classification for Fine-Grained Bug Severity Prediction

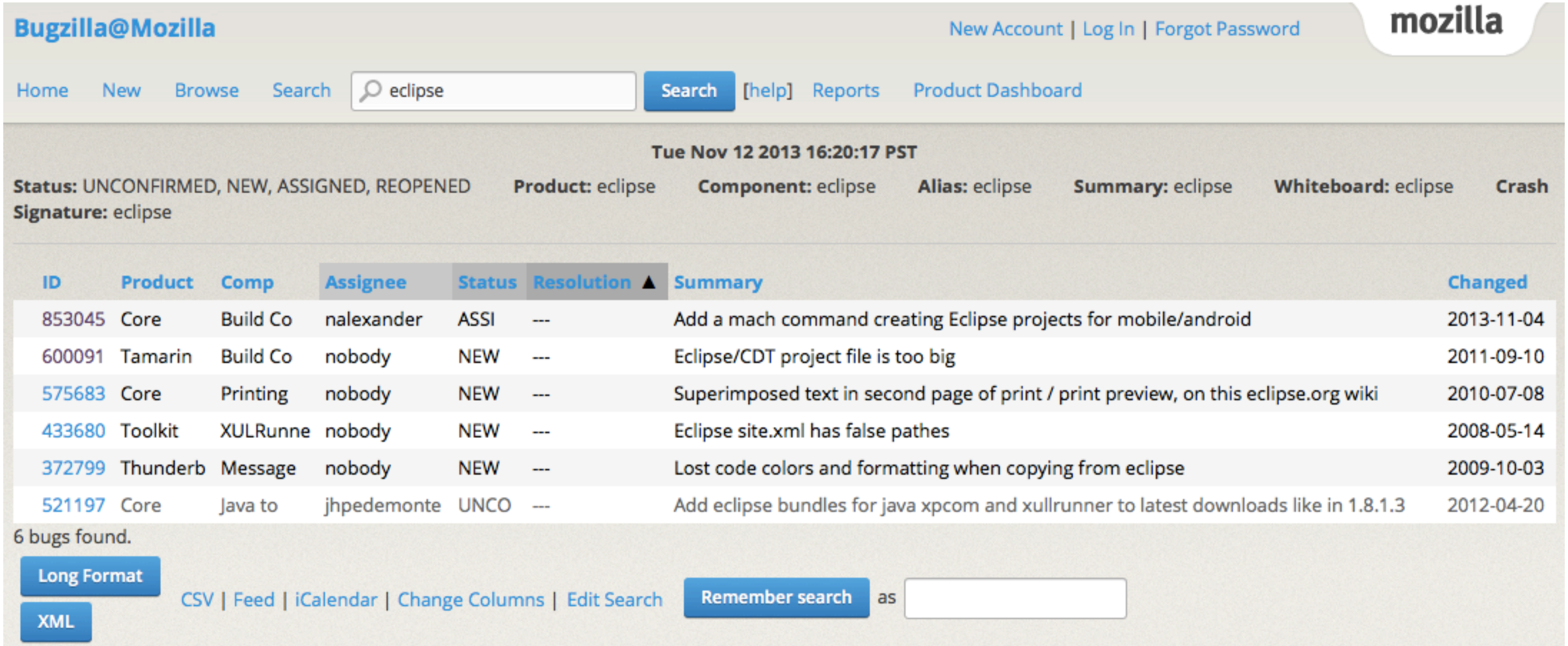
Yuan Tian, David Lo, and Chengnian Sun

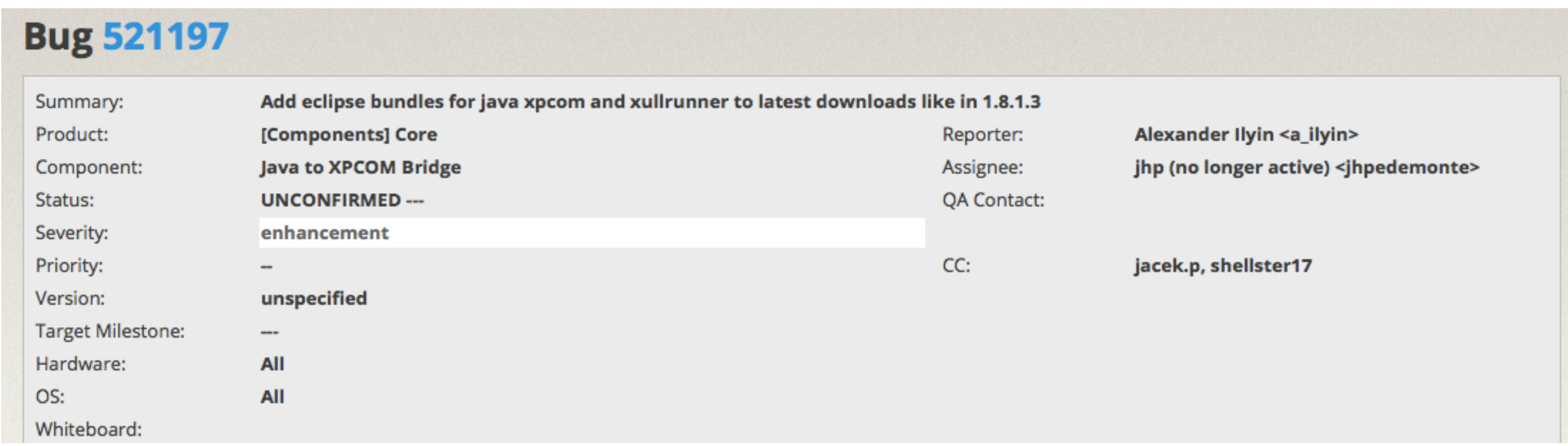
2012 19th Working Conference on Reverse Engineering

Presented by Maria Ruiz Varela and Varun Sharma

THE PROBLEM WITH AVERAGING STAR RATINGS

http://xkcd.com/937/


Bugzilla@Mozilla
New Account | Log In | Forgot Password
mozilla
Home New Browse Search eclipse Search [help] Reports Product Dashboard
Tue Nov 12 2013 16:20:17 PST
Status: UNCONFIRMED, NEW, ASSIGNED, REOPENED Product: eclipse Component: eclipse Alias: eclipse Summary: eclipse Whiteboard: eclipse Crash Signature: eclipse
ID Product Comp Assignee Status Resolution ▲ Summary Changed
853045 Core Build Co nalexander ASSI --- Add a mach command creating Eclipse projects for mobile/android 2013-11-04
600091 Tamarin Build Co nobody NEW --- Eclipse/CDT project file is too big 2011-09-10
575683 Core Printing nobody NEW --- Superimposed text in second page of print / print preview, on this eclipse.org wiki 2010-07-08
433680 Toolkit XULRunne nobody NEW --- Eclipse site.xml has false pathes 2008-05-14
372799 Thunderb Message nobody NEW --- Lost code colors and formatting when copying from eclipse 2009-10-03
521197 Core Java to jhpedemonte UNCO --- Add eclipse bundles for java xpcom and xullrunner to latest downloads like in 1.8.1.3 2012-04-20
6 bugs found.
Long Format
XML
CSV | Feed | iCalendar | Change Columns | Edit Search
Remember search as


Bug 521197
Summary: Add eclipse bundles for java xpcom and xullrunner to latest downloads like in 1.8.1.3
Product: [Components] Core
Reporter: Alexander Ilyin <a_ilyin>
Component: Java to XPCOM Bridge
Assignee: jhp (no longer active) <jhpedemonte>
Status: UNCONFIRMED ---
QA Contact:
Severity: enhancement
Priority: --
CC: jacek.p, shellster17
Version: unspecified
Target Milestone: ---
Hardware: All
OS: All
Whiteboard:

Contribution

- Fine-grained severity label prediction
- (IR)-based nearest neighbor to predict labels
- BM25F extension to measure similarity of textual information between two reports.
- Analyzed bug reports tracked in Bugzilla for Eclipse, OpenOffice, and Mozilla.

Context

- Fine grained severity prediction
  - 5 levels
- Studied bugs from Eclipse, OpenOffice, and Mozilla
- Contingent on the existence of duplicates
  - Label of duplicates are known
- Nicely structured bug reports such as Bugzilla bug tracking system

Not all reports are structured

Highly unstructured, redundant event logs from very large scale systems

- **NULL RAS BGLMASTER FAILURE ciodb exited normally with exit code 0**

- **kernel: VIPKL(1): [create_mr] MM_bld_hh_mr failed (-253:VAPI_EAure = no**

- **kernel: Uhhuh. NMI received. Dazed and confused, but trying to continue**

- **kernel: Losing some ticks… checking if CPU frequency changed.**

1. Compute similarity

$$REP(d,q) = \sum_{i=1}^{4} w_i \times feature_i$$

Linear combination of 4 features: Relevant features will have a higher score

$$\begin{aligned}
feature_1(d,q) &= BM25F_{ext}(d,q) \text{ //of unigrams} \\
feature_2(d,q) &= BM25F_{ext}(d,q) \text{ //of bigrams} \\
feature_3(d,q) &= \begin{cases} 1, & \text{if } d.prod = q.prod \\ 0, & \text{otherwise} \end{cases} \\
feature_4(d,q) &= \begin{cases} 1, & \text{if } d.comp = q.comp \\ 0, & \text{otherwise} \end{cases}
\end{aligned}$$

(1) And (2) Compute textual similarities based on two fields: Summary and description

(3) and (4) Compute non-textual similarities based on binary attributes

Background

Information Retrieval to calculate similarity between two textual documents

$$BM25F_{ext}(d,q) = \sum_{t \in d \cap q} IDF(t) \times \frac{TF_D(d,t)}{k + TF_D(d,t)} \times W_Q$$

$$W_Q = \frac{(l+1) \times TF_Q(q,t)}{l + TF_Q(q,t)}$$

Global importance of a word: Inverse document frequency

Local importance of a word: Aggregation of local importance of a word fore each field in document d

k – controls contribution of local importance to overall score

l – controls contribution of local importance of word t in document q to overall score

2. Assign label

$$\left\lfloor \frac{\sum_{i=0}^{k}(NNSet[i].Sim \times NNSet[i].Label)}{\sum_{i=0}^{k}(NNSet[i].Sim)} + 0.5 \right\rfloor$$

- Example

A bug report with top 3 neighbors , and labels 5, 4 and 3

$$REP(BQ, N_1) = 0.5$$
$$REP(BQ, N_2) = 0.45$$
$$REP(BQ, N_3) = 0.35$$

$$\begin{aligned}
\text{Label} &= \left\lfloor \frac{\sum_{i=0}^{3}(REP(BQ,N_i)\times N_i.Label)}{\sum_{i=0}^{k}(REP(BQ,N_i))} + 0.5 \right\rfloor \\
&= \left\lfloor \frac{(0.5\times5+0.45\times4+0.35\times3)}{(0.5+0.45+0.35)} + 0.5 \right\rfloor \\
&= \left\lfloor \frac{(2.5+1.8+1.05)}{1.3} + 0.5 \right\rfloor \\
&= 4
\end{aligned}$$

Experiments (Comparison Results)

Results for Mozilla:

Severity	INSPect			Severis [Offline]		
	Precision	Recall	F Measure	Precision	Recall	F Measure
blocker	33.9%	31.3%	32.6%	100%	0.2%	0.4%
critical	64.0%	67.8%	65.9%	82.6%	53.7%	65.1%
major	53.5%	55.2%	54.3%	43.9%	93.1%	59.7%
minor	38.9%	33.4%	35.9%	50.5%	1.8%	3.4%
trivial	38.4%	33.6%	35.8%	19.7%	1.1%	2.2%

F measure for blocker, critical, minor and trivial labels are improved by 8,038%, 1.2%, 957%, and 1,528% respectively over Severis

For the major label, INSPect lose out to Severis by 9.0%.

Experiments (Comparison Results)

Results for Eclipse:

Severity	INSpect			Severis [Offline]		
	Precision	Recall	F Measure	Precision	Recall	F Measure
blocker	25.2%	27.0%	26.0%	0.0%	0.0%	0.0%
critical	28.2%	29.8%	29.0%	22.3%	39.7%	28.5%
major	58.0%	57.5%	57.8%	48.2%	66.8%	56.0%
minor	42.4%	38.4%	40.3%	7.6%	0.1%	0.2%
trivial	28.2%	25.0%	26.5%	0.0%	0.0%	0.0%

F measure for blocker, critical, major, minor, and trivial labels are improved by infinity, 1.7%, 3.2%, 20,055%, and infinity, respectively over Severis.

Experiments (Varying parameter k)

The authors further investigated the effect of changing the parameter k on the overall effectiveness.

The effect of varying k (k = 1, 5, 10, 20) on F measure for OpenOffice, Mozilla, and Eclipse datasets

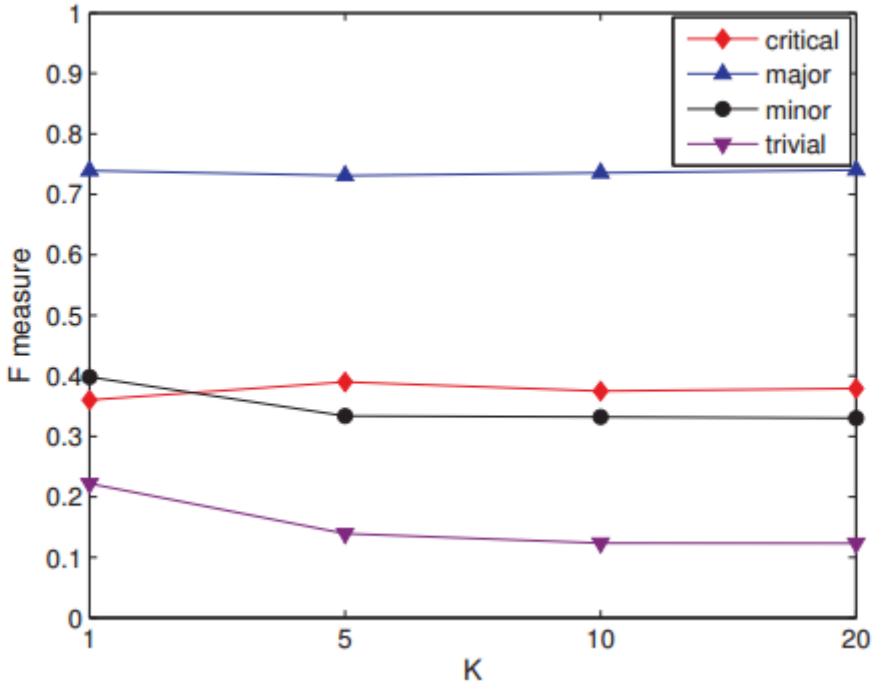
By increasing k, more nearest neighbors are considered .

Experiments (Varying parameter k)

Results for OpenOffice

- F measure of critical increases as we increase k.

- F measures of minor and trivial decrease as we increase k

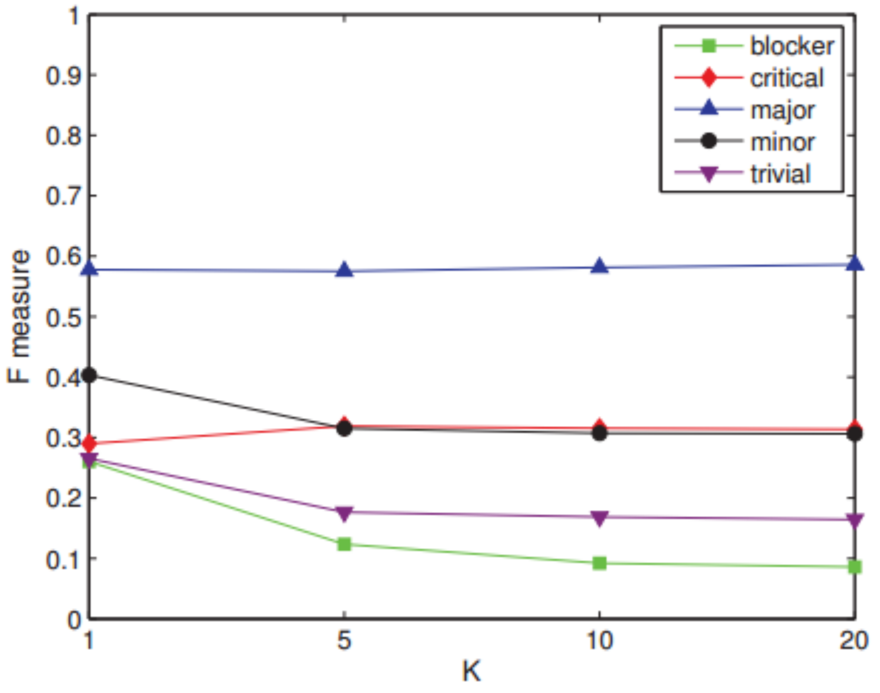


Experiments (Varying parameter k)

Results for Eclipse

- F measure of critical increases as we increase k.

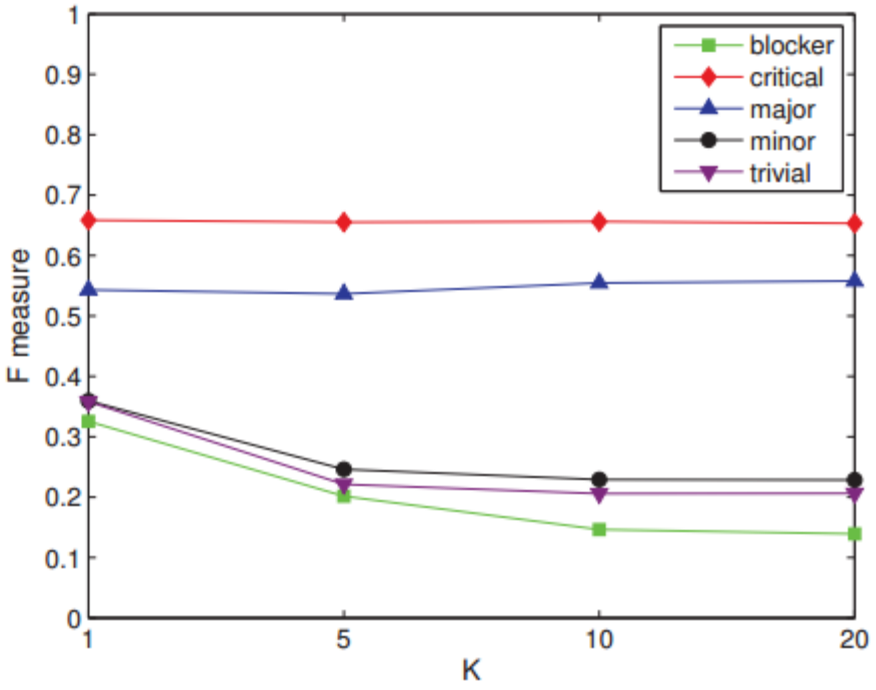
- F measures of three severity labels, blocker, minor and trivial decrease



Experiments (Varying parameter k)

Results for Mozilla

- F measure of major slightly increases as we increase k.
- F measures for three severity labels, blocker, minor, and trivial, decrease as we increase k



Experiments (Threats to Validity & Discussion)

The authors considered three threats of validity:

1. Threats to construct validity
2. Threats of internal validity
3. Threats of external validity

Experiments (Threats to Validity & Discussion)

1. Threats to construct validity

It relates to the suitability of the evaluation metrics.

2. Threats of internal validity

It refers to errors in the experiment.

Final severity labels are used as ground truth labels to measure how good they are.

Experiments (Threats to Validity & Discussion)

3. Threats of external validity

It refers to the generalizability of the findings.

The authors have considered three medium - large software systems: Eclipse, OpenOffice, and Mozilla with more than 65000 bug reports.

The three projects are written in different programming languages, and have different background and user groups.

Bug Severity Prediction

2010 (Menzies & Marcus) Pre-processing - tokenization, stop work removal and stemming
important tokens were identified using terms frequency-inverse document frequency and information gain .
classification approach - Ripper rule learner.

2008 (Lamkanfi & Demeyer)Pre-processing - tokenization, stop work removal and stemming
Classification approach - Naive Bayes classifier to predict the severity of Bugs
coarse grained bug severity labels

Analyzing Bug Reports

2007 (Runeson et al.)Calculates frequency of common words appearing in documents as a similarity measure using Natural language processing technique.

2008 (Wang et al.)Calculate Natural Language Similarities (NL-S) between new bug report and existing bug report then calculate the execution information based Similarities (E-S). Target reports are retrieved using a heuristic function.

2010 (Sun et al.)The approach uses a technique that leverages SVM for duplicate bug report detection.

Categorization of bug reports to reduce maintenance effort

2003 (Pordguski et al.) Grouped reported software failures, by using a classification strategy i that involves the use of supervised and unsupervised pppattern classification

2002 (Tamrawi et al.) The approach was for automatic bug triaging based on fuzzy set-based modeling of bug-fixing expertise of develop

Text Mining for SE

2010 (Haiduc et al.) Proposed a method to summarize source code to support program comprehension.

2003 (Marcus and Maletic) Propose an approach to link documentation to source code using Latent Semantic Indexing.

Conclusion

- Severity labels are important.
- Proposed a new approach to predict severity labels.
- Considered duplicate bug reports.
- Achieved improved values of F measure over the state-of-the-art approach on fine-grained severity label prediction.