Investigation of IR based Topic Models on Issue Tracking Systems to Infer Software-Specific Semantic Related Term Pairs

Denzil Correa[1] Sangeeta Lal[2] Ashish Sureka[3]

[1]IIIT Delhi, India (denzilc@iiitd.ac.in)

[2]JIIT Noida, India (sangeeta@jiit.ac.in)

[3]ABB India, India (ashish.sureka@in.abb.com)

IC3 2017

Table of Contents

Table of Contents

Search and Exploration of Software Repositories

Searching software depositories by developers

Software maintenance activities involve non-trivial tasks like bug fixing, feature enhancements and software reengineering

Requires searching code bases and repositories like version control systems, issue tracking systems and software documentation.

Search and exploration of large information repositories is non-trivial as the terms used to query the knowledge base could be highly domain-specific [13]

Domain Specific Search

Domain-specific semantic vocabulary gap

Developer may have little or no knowledge about the query terms to search the knowledge base to fulfill their information need [11]

Existing lexical resources in the English language like WordNet are insufficient due to domain-specific issues [28].

Need to bridge the domain-specific semantic vocabulary gap and build domain-specific lexical resources

Domain-specific lexical resource

LSI and LDA

We propose a novel approach to automatically infer semantically related terms from in a software context to facilitate the construction of a domain-specific lexical resource

We infer semantically related terms from free form natural language textual data contained in defect tracking systems by use of IR based topic models

We investigate the application of two popular and widely used IR based topic models Latent Semantic Indexing(LSI) and Latent Dirichlet Allocation(LDA)

Table of Contents

Literature Survey - I

Wang et al. [34]

Wang et al. propose an algorithm based on an holistic method of context-based and co-occurrence based term scoring to extract technical paraphrases from noisy bug report corpora [34]

Yang et al. [36]

Yang et al. propose a simple similarity based technique based on clustering comments and code to infer semantically related words in software context [36]

Literature Survey - II

Shepherd et al. [26]

Shepherd et al. propose a NLP based method to locate and understand high-level concepts in source code repositories [26].

Aggarwal et al. [2]

Aggarwal et al. present an application of mining bug report description and threaded discussion comments using Latent Dirichlet Allocation (LDA) which is a topic modeling technique [2]

Table of Contents

Novel and Unique Contributions

Infer semantically related term pairs

We propose IR based topic model approach to infer semantically related term pairs in software as an aid to construction of a domain-specific lexical resource construction

Experiments on Google Chromium

We demonstrate the efficacy of our inferred lexical resource of a large bug report repository of a popular open-source web browser, Google Chromium, on duplicate bug report detection which is one of the significant software maintenance task.

Table of Contents

Latent Semantic Indexing (LSI)

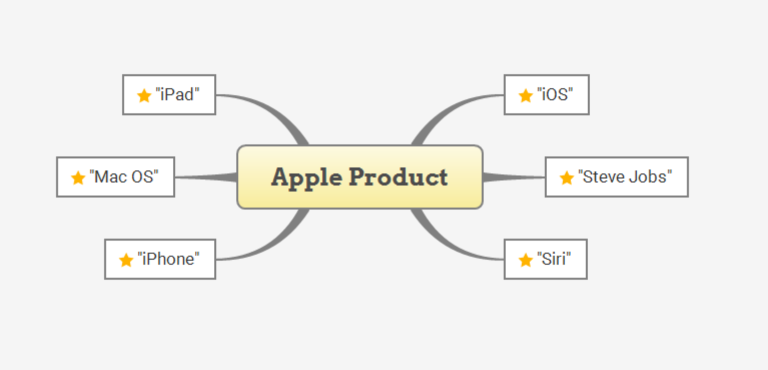


Figure: Synonym or related keyword of any primary keyword (Source: Internet)

Latent Semantic Indexing (LSI)

Information Retrieval based Method

IR based method to extract meaningful word representations from a collection of documents [8]

Objective of LSI is to infer latent relationships for words occurring within and across documents

LSI helps uncover words with similar meanings (synonymy) and words which may have more than one meaning (polysemy).

Table of Contents

Latent Dirichlet Allocation (LDA)

Topic 1		Topic 2		Topic 3	
term	**weight**	**term**	**weight**	**term**	**weight**
game	0.014	space	0.021	drive	0.021
team	0.011	nasa	0.006	card	0.015
hockey	0.009	earth	0.006	system	0.013
play	0.008	henry	0.005	scsi	0.012
games	0.007	launch	0.004	hard	0.011

Figure: Topic modeling with LDA (Source: Internet Databricks)

Latent Dirichlet Allocation (LDA)

Information Retrieval based Method

Probabilistic generative model that helps discovers document-topic and topic-word distributions from a collection of documents [7]

The words occurring in the same topic of the output of LDA are connected to each other via a semantic relationship.

Issue Tracking System Dataset

Download Date	August 12, 2012
Number of Reports Available in Issue Tracker	142,175
Number of Reports Available for Download	134,410
Number of Reports excluding WontFix, Unconfirmed, Invalid, Untriaged, Unknown, FixUnreleased	90,431
Number of *Open* Reports	29,596
Number of *Closed* Reports	104,814
Number of *Duplicate* Reports	28,460
Number of *Duplicate* Reports with Master report available	20,936
Timestamp of first bug report	2008-08-30 16:00:21
Timestamp of last bug report	2012-08-11 21:16:59

Table: Experimental Dataset Details for Google Chromium DTS

Dataset Details

Publicly Available Google Chromium Dataset

Google Chromium software, a popular open-source web browser, which contains 134,000+ bug reports.[a]

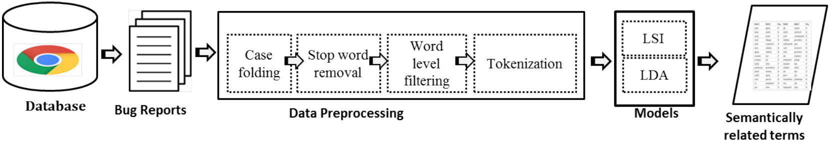
We had 142,175 bug reports available but only 134,410 bug reports were available for download

We exclude reports which are marked as - *WontFix*, *Unconfirmed*, *Invalid*, *Untriaged*, *Unknown* and *FixUnreleased*.

[a] http://crbug.com/

Table of Contents

Solution Approach and Framework



Bug report extraction, text pre-processing, application of LSI and LDA, extraction of semantically related terms

Table of Contents

Multi-Step Pre-processing - I

Case Folding

We convert all the terms in the DTS corpora into lower case

Stopword Removal

We use two types of stop word removal techniques. We use a standard stopword list in the English language.[a] Due to the noisy nature of DTS corpora, we consider the top 30 frequently occurring words in the corpus also as stopwords

[a] http://www.link-assistant.com/seo-stop-words.html

Multi-Step Pre-processing - II

Word Level Filtering

We eliminate terms which are lesser than 3 characters in length

Tokenization

We use a white space tokenizer to form tokens to our IR based topic models

Table of Contents

Libraries and Tools Used

Latent Semantic Indexing (LSI)

We used the implementation from Gensim, python framework to perform our experiments.[a]

[a] http://radimrehurek.com/gensim/

Latent Dirichlet Allocation (LDA)

We used the implementation from the Stanford Topic Modeling Toolbox for our experiments.[a]

[a] http://nlp.stanford.edu/software/tmt/tmt-0.4/

Automatically Inferred Semantically Related Terms - I

Term Pairs	Context
ssl – certificate	Software
omnibox – search	Google Chromium
html – background	Google Chromium
css – element	Google Chromium
int–bool	Code
com.google.chrome.framework	Code
std:allocator – memcheck	Code
renderblock.cpp – resourcedispatcher	Code

Automatically Inferred Semantically Related Terms - II

Term Pairs	Context
time – event	English
flaky – failure	English
error – failure	English
window – tab	Software
mouse – drag	Software
hash – base/message	Software
ssl – certificate	Software

Successful Extraction of Semantically Retated Terms - I

English Context

Term pairs *time* – *event* and *error* – *failure* are semantically similar in the English language. In other words, these terms are synonyms in the English dictionary

General Software Context

The term pairs *window* – *tab* and *ssl* – *certificate* are semantically similar terms in a general software context on web browsers. SSL commonly used in the form security certificates in web browsing to encrypt communication by web browsers

Successful Extraction of Semantically Retated Terms - II

Google Chromium Context

The term pairs *omnibox – search* and *css – element* are semantically related in the Google Chromium software context in particular.

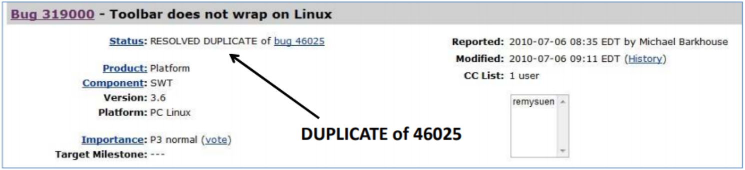
Code Context

int – bool are date types in code which are related to each other as they have the same data type. In addition, *std:allocator – mem-check* is a library and function definitions which are used for memory leakage checks.

Table of Contents

Duplicate Bug Report Detection - I

Snapshot of Eclipse Project BUG ID: 319000


Bug 319000 - Toolbar does not wrap on Linux

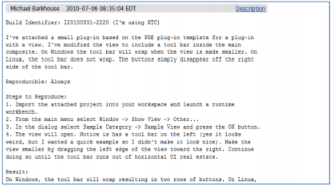
Status: RESOLVED DUPLICATE of bug 46025

Product: Platform
Component: SWT
Version: 3.6
Platform: PC Linux

Importance: P3 normal (vote)
Target Milestone: ---

Reported: 2010-07-06 08:35 EDT by Michael Barkhouse
Modified: 2010-07-06 09:11 EDT (History)
CC List: 1 user

remysuen


Michael Barkhouse 2010-07-06 08:35:04 EDT Description

Build Identifier: I20100331-2220 (I'm using RTC)

I've attached a small plug-in based on the PDE plug-in template for a plug-in with a view. I've modified the view to include a tool bar inside the main composite. On Windows the tool bar will wrap when the view is made smaller. On Linux, the tool bar does not wrap. The buttons simply disappear off the right side of the tool bar.

Reproducible: Always

Steps to Reproduce:
1. Import the attached project into your workspace and launch a runtime workbench.
2. From the main menu select Window -> Show View -> Other...
3. In the dialog select Sample Category -> Sample View and press the OK button.
4. The view will open. Notice is has a tool bar on the left (yes it looks weird, but I wanted a quick example so I didn't make it look nice). Make the view smaller by dragging the left edge of the view toward the right. Continue doing so until the tool bar runs out of horizontal UI real estate.

Result:
On Windows, the tool bar will wrap resulting in two rows of buttons. On Linux,

BUG DESCRIPTION

Duplicate Bug Report Detection - II

Use-case to demonstrate the efficacy of our approach

Experimental dataset contains 20,936 duplicate bug reports. We only use the description of the bug report for our task.

Procedure to Compute Similarity

The output of LSI and LDA is a term-term pair each consisting of a similarity score. We iterate through each document in our experimental duplicate bug report dataset and compare the similarity with all the other bug reports in the corpus by using these scores from LSI and LDA.

Table of Contents

Recall Rate as Evaluation Metrics

Top-K list	LSI Recall Rate	LDA Recall Rate
10	44.14	34.32
20	49.12	35.12
30	53.47	37.73
40	55.62	40.51
50	57.72	40.97

Table: Recall Rates of IR based topic models LSI and LDA on various top-k list sizes for duplicate bug report detection task.

Experimental Result Analysis

Key Takeaways

LSI performs significantly better than LDA while the recall rates achieved are not significantly high.

There are various factors which affect the recall rates and this may not be specifically due to the nature of our lexical resource

Conclusion - I

We investigate the effectiveness of IR based topic models like LSI and LDA to infer semantically related term pairs in software domain context

We apply our technique on Google Chromium DTS, an open source popular browser, and are able to infer semantically related term pairs in various contexts like English language, web browser, Google Chromium specific and code snippets.

Conclusion - II

We use the automatically inferred semantic term pairs (as is) to look into duplicate bug report detection

Initial experiments suggest that IR based topic models are an interesting direction to pursue for construction of lexical resources.

References I

[1] Ieee std. 1219 : Standard for software maintenance. In *IEEE Computer Society Press*, 1993.

[2] Ayushi Aggarwal, Gajendra Waghmare, and Ashish Sureka. Mining issue tracking systems using topic models for trend analysis, corpus exploration, and understanding evolution. In *Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*, RAISE 2014, pages 52–58, 2014.

[3] Giuliano Antoniol, Kamel Ayari, Massimiliano Di Penta, Foutse Khomh, and Yann-Gaël Guéhéneuc. Is it a bug or an enhancement?: a text-based approach to classify change requests. In *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*, CASCON '08, pages 23:304–23:318, New York, NY, USA, 2008. ACM.

[4] John Anvik, Lyndon Hiew, and Gail C. Murphy. Who should fix this bug? In *Proceedings of the 28th international conference on Software engineering*, ICSE '06, pages 361–370, New York, NY, USA, 2006. ACM.

References II

[5] Olga Baysal, Ian Davis, and Michael W. Godfrey. A tale of two browsers. In *Proceedings of the 8th Working Conference on Mining Software Repositories*, MSR '11, pages 238–241, New York, NY, USA, 2011. ACM.

[6] Dane Bertram, Amy Voida, Saul Greenberg, and Robert Walker. Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, CSCW '10, pages 291–300, New York, NY, USA, 2010. ACM.

[7] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[8] Scott Deerwester, Susan T. Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[9] Christiane Fellbaum. Wordnet. *Theory and Applications of Ontology: Computer Applications*, pages 231–243, 2010.

References III

[10] Chiara Francalanci and Francesco Merlo. Empirical analysis of the bug fixing process in open source projects. In Barbara Russo, Ernesto Damiani, Scott A. Hissam, Björn Lundell, and Giancarlo Succi, editors, *OSS*, volume 275 of *IFIP*, pages 187–196. Springer, 2008.

[11] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, 1987.

[12] Michael Gegick, Pete Rotella, and Tao Xie. Identifying security bug reports via text mining: An industrial case study. In *Proc. 7th Working Conference on Mining Software Repositories (MSR 2010)*, pages 11–20, May 2010.

[13] Emily Hill. *Integrating natural language and program structure information to improve software search and exploration*. University of Delaware, 2010.

[14] Nicholas Jalbert and Westley Weimer. Automated duplicate detection for bug tracking systems. In *The 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2008, June 24-27, 2008, Anchorage, Alaska, USA, Proceedings*, pages 52–61. IEEE Computer Society, 2008.

References IV

[15] Don H Johnson, Sinan Sinanovic, et al. Symmetrizing the kullback-leibler distance. *IEEE Transactions on Information Theory*, 1(1):1–10, 2001.

[16] Nilam Kaushik and Ladan Tahvildari. A comparative study of the performance of ir models on duplicate bug detection. In *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, pages 159–168. IEEE, 2012.

[17] Foutse Khomh, Brian Chan, Ying Zou, and Ahmed E. Hassan. An entropy evaluation approach for triaging field crashes: A case study of mozilla firefox. In *Proceedings of the 2011 18th Working Conference on Reverse Engineering*, WCRE '11, pages 261–270, Washington, DC, USA, 2011. IEEE Computer Society.

[18] Bennet P. Lientz, E. Burton Swanson, and Gail E Tompkins. Characteristics of application software maintenance. *Communications of the ACM*, 21(6):466–471, 1978.

References V

[19] Chao Liu, Xifeng Yan, Long Fei, Jiawei Han, and Samuel P Midkiff. Sober: statistical model-based bug localization. *ACM SIGSOFT Software Engineering Notes*, 30(5):286–295, 2005.

[20] David M Nichols and Michael B Twidale. Usability processes in open source projects. *Software Process: Improvement and Practice*, 11(2):149–162, 2006.

[21] Lucas D. Panjer. Predicting eclipse bug lifetimes. In *Proceedings of the Fourth International Workshop on Mining Software Repositories*, MSR '07, pages 29–, Washington, DC, USA, 2007. IEEE Computer Society.

[22] Thomas M Pigoski. *Practical software maintenance: best practices for managing your software investment*. John Wiley & Sons, Inc., 1996.

[23] Per Runeson, Magnus Alexandersson, and Oskar Nyholm. Detection of duplicate defect reports using natural language processing. In *Software Engineering, 2007. ICSE 2007. 29th International Conference on*, pages 499–510. IEEE, 2007.

References VI

[24] Holger Schackmann, Henning Schaefer, and Horst Lichter. Evaluating process quality based on change request data — an empirical study of the eclipse project. In *Proceedings of the International Conferences on Software Process and Product Measurement*, IWSM '09 /Mensura '09, pages 227–241, Berlin, Heidelberg, 2009. Springer-Verlag.

[25] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5:3–55, January 2001.

[26] David Shepherd, Zachary P Fry, Emily Hill, Lori Pollock, and K Vijay-Shanker. Using natural language program analysis to find and understand action-oriented concerns. In *Int. Conf. on Aspect-oriented Software Development*, 2007.

[27] David Shepherd, Lori Pollock, and K Vijay-Shanker. Towards supporting on-demand virtual remodularization using program graphs. In *Proceedings of the 5th international conference on Aspect-oriented software development*, pages 3–14. ACM, 2006.

References VII

[28] Giriprasad Sridhara, Emily Hill, Lori Pollock, and K Vijay-Shanker. Identifying word relations in software: A comparative study of semantic similarity tools. In *Program Comprehension, 2008. ICPC 2008. The 16th IEEE International Conference on*, pages 123–132. IEEE, 2008.

[29] Ashish Sureka. Learning to classify bug reports into components. *Objects, Models, Components, Patterns*, pages 288–303, 2012.

[30] Ashish Sureka. Learning to classify bug reports into components. In Carlo A. Furia and Sebastian Nanz, editors, *TOOLS (50)*, volume 7304 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2012.

[31] Ashish Sureka and Pankaj Jalote. Detecting duplicate bug report using character n-gram-based features. In *Proceedings of the 2010 Asia Pacific Software Engineering Conference*, APSEC '10, pages 366–374, Washington, DC, USA, 2010. IEEE Computer Society.

References VIII

[32] Michael B. Twidale and David M. Nichols. Exploring usability discussions in open source development. In *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences - Volume 07*, HICSS '05, pages 198.3–, Washington, DC, USA, 2005. IEEE Computer Society.

[33] Huaiqing Wang and Chen Wang. Open source software adoption: A status report. *IEEE Softw.*, 18:90–95, March 2001.

[34] Xiaoyin Wang, David Lo, Jing Jiang, Lu Zhang, and Hong Mei. Extracting paraphrases of technical terms from noisy parallel software corpora. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 197–200. Association for Computational Linguistics, 2009.

[35] Xiaoyin Wang, Lu Zhang, Tao Xie, John Anvik, and Jiasu Sun. An approach to detecting duplicate bug reports using natural language and execution information. In *Proceedings of the 30th international conference on Software engineering*, ICSE '08, pages 461–470, New York, NY, USA, 2008. ACM.

References IX

[36] Jinqiu Yang and Lin Tan. Inferring semantically related words from software context. In *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*, pages 161–170. IEEE, 2012.

[37] Shahed Zaman, Bram Adams, and Ahmed E. Hassan. Security versus performance bugs: a case study on firefox. In *Proceedings of the 8th Working Conference on Mining Software Repositories*, MSR '11, pages 93–102, New York, NY, USA, 2011. ACM.

[38] Shahed Zaman, Bram Adams, and Ahmed E. Hassan. A qualitative study on performance bugs. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR)*, Zurich, Switzerland, June 2012.

[39] Jian Zhou, Hongyu Zhang, and David Lo. Where should the bugs be fixed? - more accurate information retrieval-based bug localization based on bug reports. In *Proceedings of the 2012 International Conference on Software Engineering*, ICSE 2012, pages 14–24, Piscataway, NJ, USA, 2012. IEEE Press.