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

Denzil Correa
IIIT-D,
Delhi, India
denzilc@iiitd.ac.in

Sangeeta Lal
Jaypee Institute of Information Technology,
Noida, Uttar-Pradesh, India
sangeeta@jiit.ac.in

Ashish Sureka
ABB Corporate Research,
Bangalore, India
ashish.sureka@in.abb.com

***Abstract*—Software maintenance is a core component of any software development life-cycle. Contemporary software systems contain voluminous and complex information stored in software repositories. Software maintenance professionals spend significant amount of time in search and exploration of these repositories for common maintenance tasks like bug fixing, feature enhancements, code refactoring and reengineering. Therefore, tools and methods to facilitate search in software repositories can aid software maintenance professionals to have faster access to required information and increase productivity. A domain-specific lexical resource is an important tool to bridge the semantic gap existing between the information need and search query. In this work, we investigate the use of information retrieval (IR) based topic models (like LSI and LDA) to infer semantically related terms for a software context specific lexical resource. We perform our experiments on Google Chromium – a widely popular open-source browser – issue tracker system which contains 134,000+ bug reports. We divide our study into two parts – (1) In the first part, we apply our IR models on free form natural language textual data present in defect tracking systems. We perform qualitative analysis on the obtained output and uncover semantically related terms in the Google Chromium software context. We observe that we are able to infer semantically similar term pairs in four different contexts of English language, Software, Google Chromium and Code details. (2) In second part of this study, we utilize the semantically inferred terms obtained from the output of IR models to facilitate the software maintenance task of duplicate bug report detection. Our results demonstrate that the use of IR based topic models on defect tracking systems to automatically infer semantically related terms can help build a software domain-specific lexical resource and reduce the vocabulary gap.**

***Index Terms*—Automated Software Engineering, Issue Tracking System, Latent Dirichlet Allocation (LDA), Latent Semantic Indexing (LSI), Mining Software Repositories, Software Maintenance**

I. Introduction

IEEE Standard 1219 defines software maintenance as – *The modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment* [1]. Software Maintenance is a fundamental activity of any software development process. Typical software maintenance activities involve non-trivial tasks like bug fixing, feature enhancements and software reengineering which require the software maintenance professionals to search code bases and other software repositories (or knowledge bases) like version control systems, issue tracking systems and software documentation. Modern day software systems have evolved over decades and hence are large, complex and continue to grow every day. Thus, software maintenance professionals spend a significant amount of time in information search of code bases and other software repositories. Previous work also shows that more than 80% of the time in software maintenance phase of a software process is spent on tasks other than bug fixing [2]. These activities include tasks like duplicate bug report detection and bug triaging which require significant time and effort towards information search from one or more software repositories. Search and exploration of large information repositories is non-trivial as the words used to query the knowledge base could be highly domain-specific [3]. A software maintenance professional may have little or no knowledge about the query terms to search the knowledge base to fulfill his information need [4]. Existing lexical resources in the English language like WordNet are insufficient due to domain-specific issues [5]. Hence, there is a need to bridge the domain-specific semantic vocabulary gap and build domain-specific lexical resources to assist software maintenance professionals in search and exploration of software knowledge and code bases.

In this work, 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. In particular, we focus our attention to infer semantically related terms from free form natural language textual data contained in defect tracking systems by use of IR based topic models. Defect tracking systems (DTS), also called issue tracking systems, are used to manage issues related to software like bug fixes and feature enhancements. DTS contains detailed information on the software and therefore, are great knowledge bases to extract useful information. Moreover, many time consuming and sophisticated software maintenance tasks like duplicate bug report detection, bug triaging, product component assignment and bug localization require search and exploration of DTS [6]–[8]. Therefore, we investigate the

application of two popular and widely used IR based topic models Latent Semantic Indexing(LSI) and Latent Dirichlet Allocation(LDA) to infer semantically related terms in the software domain. We perform our experiments on Google Chromium DTS – a widely popular open source browser – consisting of 134,000+ reports, and present a qualitative analysis of the semantically inferred terms. We also use the lexical resource obtained from our experiments to aid the problem of *duplicate bug report detection* which is one of the important software maintenance activity. Our experimental results show that the application of IR based models on DTS can be an effective complimentary solution to build software domain specific lexical resources.

II. Background and Research Contributions

In this section, we review the closely related work to our research and list down our specific research contributions.

A. Prior Work

The inference of semantically similar terms in software context is a significant and technically challenging task [5]. 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 [9]. They evaluate their approach on *OpenOffice* bug reports, a popular open-source document editing software, and report promising results. Yang *et al.* propose a simple similarity based technique based on clustering comments and code to infer semantically related words in software context [10]. They evaluate their technique on several open source projects like The Linux Kernel and jBidWatcher written in C and Java and report reasonable accuracy. Shepherd *et al.* propose a NLP based method to locate and understand high-level concepts in source code repositories [11]. They use query expansion with word recommendation techniques to locate concepts while an action-oriented identifier graph(AOIG) visual representation is used to help the programmer understand the concept after location. Aggarwal et al. [12] use trend analysis in issue tracking system. To the best of our knowledge, the aforementioned studies are the only closely related work to our research.

B. Research Contributions

In context to the above related work, we make the following research contributions:

1) 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.
2) We demonstrate the efficacy of our inferred lexical resource on 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.

III. Information Retrieval Based Topic Models

In this section, we briefly review our choice of IR based topic models we use to infer the semantically related terms.

A. Latent Semantic Indexing (LSI)

Latent Semantic Indexing (LSI) is an IR based method to extract meaningful word representations from a collection of documents [13]. The objective of LSI is to infer latent relationships for words occurring within and across documents. First, the collection of documents are converted to a term frequency-inverse document frequency (*tf-idf*) matrix which contains a count of terms in the respective documents. Next, the *tf-idf* matrix is reduced to a lower dimension matrix by the use of a matrix decomposition technique called Singular Value Decomposition (SVD). The number of dimensions to be reduced to is a user input parameter with no particular exact scientific method of selection. The reduced lower dimensionality matrix consists of word collocations which tend to have similar meanings due to co-occurrence in similar contexts. In particular, LSI helps uncover words with similar meanings (synonymy) and words which may have more than one meaning (polysemy).

B. Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) is a probabilistic generative model that helps discovers document-topic and topic-word distributions from a collection of documents [14]. LDA hypothesizes that each document is a distribution mixture of multiple topics and each topic is a distribution mixture of multiple words. LDA uses a bayesian generative model to create the document-topic and topic-word distributions from the collection of documents. The topic distribution in LDA is assumed to have a Dirichlet prior. The number of topics to the LDA algorithm is a user input parameter with no particular exact scientific method of selection. The words occurring in the same topic of the output of LDA are connected to each other via a semantic relationship.

IV. Experiments

In this section, we outline our experiments including dataset, setup and the semantically inferred terms. Figure IV shows steps of our experimental framework. We first extract the target dataset from Google Chromium project. We then apply some basic preprocessing steps to clean our data. We finally apply LSI and LDA models to find semantically related terms. Following subsection describe each step in detail.

A. Experimental Dataset

We now discuss our dataset used to conduct our experiments. Our research objective is to infer semantically related terms in software context to aid the construction of a domain-specific lexical resource. We exploit the information content in the DTS to extract and form these semantically related terms. We use the DTS from the Google Chromium software, a popular open-source web browser, which contains 134,000+ bug reports.[1] During the time of dataset download (August 12, 2012), we had 142,175 bug reports available but only 134,410 bug reports were available for download. Amongst these, we

[1] http://crbug.com/

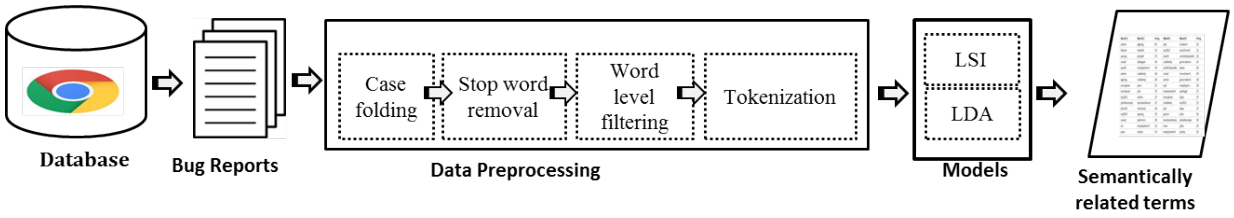

Fig. 1. Experimental framework followed in this work

exclude reports which are marked as - *WontFix*, *Unconfirmed*, *Invalid*, *Untriaged*, *Unknown* and *FixUnreleased*. We now have 90,436 reports in our dataset on which we employ our IR based topic models (LSI and LDA) on this dataset to infer semantically related terms. Table I shows the details of our experimental 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 I
EXPERIMENTAL DATASET DETAILS FOR GOOGLE CHROMIUM DTS

B. Experimental Setup

We now discuss the experimental setup used in our experiments.

1) Preprocessing: The natural language free form textual data present in DTS is noisy and requires some cleaning. We perform the following preprocessing steps

- *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.[2] Due to the noisy nature of DTS corpora, we consider the top 30 frequently occurring words in the corpus also as stopwords.
- *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.

2) Latent Semantic Indexing (LSI): The LSI algorithm requires the number of dimensions as input user parameter. The selection of this parameter is an open research question and depends on the end user application. We experiment with various dimensions and decide the number of topics to be 80. We used the implementation from *Gensim*, python framework to perform our experiments.[3]

[2]http://www.link-assistant.com/seo-stop-words.html
[3]http://radimrehurek.com/gensim/

3) Latent Dirichlet Allocation (LDA): The LDA algorithm requires the number of topics as an input user parameter. The selection of this parameter is also an open research question and depends on the end user application. We experiment with various dimensions and decide the number of topics to be 20. We also make a choice of Variational Inference Sampling as it converges faster. We used the implementation from the Stanford Topic Modeling Toolbox for our experiments.[4]

C. Automatically Inferred Semantically Related Terms

We apply both our IR based topic models on the Google Chromium DTS experimental dataset. We can see that we are able to automatically infer semantically related terms in various contexts. Table III shows some of the terms inferred from our IR based models. We now discuss some of these terms in various contexts.

Semantically Similar 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
omnibox – search	Google Chromium
html – background	Google Chromium
css – element	Google Chromium
int–bool	Code
libsystem.b.dylib – com.google.chrome.framework	Code
std:allocator – memcheck	Code
renderblock.cpp – resourcedispatcher	Code

TABLE II
AUTOMATICALLY INFERRED SEMANTICALLY RELATED TERM PAIRS FROM IR BASED TOPIC MODELS LIKE LSI AND LDA

1) English Context: The 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.

2) General Software Context: The term pairs *window – tab* and *ssl – certificate* are semantically similar terms in a general software context on web browsers. SSL is an acronym for Secure Sockets Layer and are commonly used in the form security certificates in web browsing to encrypt communication by web browsers. A browser window can have multiple tabs open and therefore are semantically related.

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

3) Google Chromium Context: The term pairs *omnibox – search* and *css – element* are semantically related in the Google Chromium software context in particular. The term *omnibox* is unique to the Google Chromium browser project and is used to denote the address bar which also doubles up as a search box ; it doesn't feature in other browsers like Mozilla Firefox and Internet Explorer.[5]

4) Code Context: Bug reports in DTS have information of execution stack traces which contain code snippets.[6] Due to such information present in the DTS the IR based topic models, are able to infer semantic term pairs even in code. For example, *int – bool* are date types in code which are related to each other as they have the same data type. In addition, *std:allocator – memcheck* is a library and function definitions which are used for memory leakage checks. *std:allocator* is a memory based class template in C++ which is used to define memory models in the Standard Template Library(STL).[7] Also, *renderblock.cpp* contains of a resource dispatcher which is used to handle inter-process communications to take care of resource requests.

The above examples show that the application of IR based topic models on bug reports obtained from DTS can help to discover semantically related term pairs in various contexts. This can help aid the construction of a software domain specific and a software project domain specific lexical resource.

D. Software Maintenance Application – Duplicate Bug Report Detection

We now use the terms inferred from our IR based topic models on an important software maintenance task in duplicate bug report detection in order to demonstrate the efficacy of our approach. Our experimental dataset contains 20,936 duplicate bug reports as shown in Table I. In our experiments, we only use the description of the bug report for our task.

1) Latent Semantic Indexing (LSI) : The output of LSI 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. The document combinations with the best similarity would be the one with the highest similarity scores. The number of dimensions is an input parameter. We conduct various experiments to fine tune the parameter and decide the number of dimensions to be 80.

2) Latent Dirichlet Allocation (LDA) : In case of LDA, we iterate through each document in our experimental duplicate bug report dataset and compare the probability distribution of the word-topic distributions. We use symmetric Kullback-Leibler (KL) divergence as our metric to denote similarity as [15]. The document combinations with the best similarity would be the one with the highest symmetric KL divergence scores. The number of topics is an input parameter. We conduct various experiments to decide the parameter and decide on the number of topics to be 20.

3) Evaluation: We use *Recall Rate* as used by Runeson *et al.* as our evaluation metric [16]. It denotes the number of duplicate bug reports detected for top-k results.

4) Results: Table shows the results of our experiments for duplicate bug report detection via both IR based models – LSI and LDA. The top-k list sizes refer to the maximum rank limit to which the master duplicate bug report is retrieved while having the child duplicate bug report as a query.

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 III
RECALL RATES OF IR BASED TOPIC MODELS LSI AND LDA ON VARIOUS TOP-K LIST SIZES FOR DUPLICATE BUG REPORT DETECTION TASK.

The recall rates show that LSI performs significantly better than LDA while the recall rates achieved are not significantly high. However, there are various factors which affect the recall rates and this may not be specifically due to the nature of our lexical resource.

V. LIMITATIONS, FUTURE WORK AND CONCLUSION

Software maintenance is a time consuming and expensive activity in most software processes. Software maintenance professionals spend significant amount of time in various program comprehension tasks by using search and exploration of code and bug report databases. The construction of lexical resources for aiding software maintenance professionals to improve their productivity is an important research problem. In this work, 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. We use the automatically inferred semantic term pairs (as is) to look into duplicate bug report detection – a significant software maintenance activity. Initial experiments suggest that IR based topic models are an interesting direction to pursue for construction of lexical resources. Recall rates for the duplicate bug report detection task are not particularly high while using the semantically related term pairs as is. This shows that despite the term-pairs extracted are useful in various contexts more work is required to adapt it to a level where it can be directly used as a lexical resource. In future, we plan to look deeper into these IR based models to automatically construct lexical resources from such semantically related term-pairs.

[5]https://support.google.com/chrome/bin/answer.py?hl=en&answer=95440

[6]http://www.cplusplus.com/reference/memory/allocator/

[7]http://www.cplusplus.com/reference/memory/allocator/

REFERENCES

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

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

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

[4] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais, "The vocabulary problem in human-system communication," *Communications of the ACM*, vol. 30, no. 11, pp. 964–971, 1987.

[5] G. Sridhara, E. Hill, L. 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*. IEEE, 2008, pp. 123–132.

[6] A. Sureka, "Learning to classify bug reports into components," *Objects, Models, Components, Patterns*, pp. 288–303, 2012.

[7] A. Sureka and P. Jalote, "Detecting duplicate bug report using character n-gram-based features," in *Proceedings of the 2010 Asia Pacific Software Engineering Conference*, ser. APSEC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 366–374.

[8] C. Liu, X. Yan, L. Fei, J. Han, and S. P. Midkiff, "Sober: statistical model-based bug localization," *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 5, pp. 286–295, 2005.

[9] X. Wang, D. Lo, J. Jiang, L. Zhang, and H. Mei, "Extracting paraphrases of technical terms from noisy parallel software corpora," in *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Association for Computational Linguistics, 2009, pp. 197–200.

[10] J. Yang and L. Tan, "Inferring semantically related words from software context," in *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*. IEEE, 2012, pp. 161–170.

[11] D. Shepherd, Z. P. Fry, E. Hill, L. 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.

[12] A. Aggarwal, G. Waghmare, and A. 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*, ser. RAISE 2014. New York, NY, USA: ACM, 2014, pp. 52–58.

[13] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.

[14] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[15] D. H. Johnson, S. Sinanovic *et al.*, "Symmetrizing the kullback-leibler distance," *IEEE Transactions on Information Theory*, vol. 1, no. 1, pp. 1–10, 2001.

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