Exploring Topic Models in Software Engineering Data Analysis: A Survey

Xiaobing Sun*, Xiangyue Liu*, Bin Li*, Yucong Duan§ Hui Yang*, Jiajun Hu*
*School of Information Engineering, Yangzhou University, Yangzhou, China
§Hainan University, Haikou, China

***Abstract*—Topic models are shown to be effective to mine unstructured software engineering (SE) data. In this paper, we give a simple survey of exploring topic models to support various SE tasks between 2003 and 2015. The survey results show that there is an increasing concern in this area. Among the SE tasks, source code comprehension and software history comprehension are the mostly studied, followed by software defects prediction. However, there is still only a few studies on other SE tasks, such as feature location and regression testing.**

***Index Terms*—Software engineering, topic models, survey**

I. Introduction

With the evolution of software, a volume of data is produced in various software repositories. Among these data, it is estimated that between 80% and 85% of the data in software repositories is unstructured [25], for example, source code, documentation, test cases, bug repositories, etc. [46]. Mining these unstructured software repositories can uncover interesting and actionable information to support various software engineering (SE) tasks, such as program comprehension, feature location, and traceability link recovery [12, 17, 46].

Among existing studies, topic models have gained an increasing concern from the SE community to help mine these unstructured software data [46]. These topic models can help develop new ways to search, explore, browse, and model large archives of texts (unstructured data). In the SE community, topic models have been used to support various SE activities [46]. In this paper, we conduct a survey of how topic models have thus far been applied in SE tasks from four SE journals and eleven conference proceedings. The survey results show that there is an increasing concern in exploring topic models in various SE tasks, such as source code comprehension, feature location, traceability link recovery, refactoring, software testing, developer recommendation, software defects prediction, software history comprehension. Among them, source code comprehension and software histories comprehension are the mostly studied.

II. Topic Models

Topic models were originated from the field of natural language processing and information retrieval (IR) to index, search, and cluster a large amount of unstructured and unlabeled documents [9]. The topic models explore the documents by representing them with topics, which are a collection of terms that co-occur frequently in the documents.

Latent Dirichlet Allocation (LDA) is one of the most popular probabilistic topic models [11], which models each document as a mixture of K corpus-wide topics, and each topic as a mixture of the terms in the corpus. More specifically, it means that there is a set of topics to describe the entire corpus, where each document can contain more than one of these topics, and each term in the entire repository can be contained in more than one of these topics. Hence, LDA is able to discover a set of ideas or themes that well describe the entire corpus.

Several variants of LDA have been proposed, including Author-Topic Model [56], Dynamic Topic Model [10], Relational Topic Models [13], On-Line LDA [2], Pachinko Allocation Model (PAM) [37], citation influence model [16], etc. All of these variants applied additional constraints on the basic LDA model in some way. More details of these topic model variants can refer to these references.

Table I
Survey venue and statistics

Venue	Full name	Count
ICSE	International Conference on Software Engineering	4
CSMR-WCRE / SANER	International Conference on Software Maintenance, Reengineering, and Reverse Engineering / International Conference on Software Analysis, Evolution, and Reengineering	3
ICSM / ICSME	International Conference on Software Maintenance / International Conference on Software Maintenance and Evolution	3
ICPC	International Conference on Program Comprehension	3
ASE	International Conference on Automated Software Engineering	3
ISSRE	International Symposium on Software Reliability Engineering	2
MSR	International Working Conference on Mining Software Repositories	8
OOPSLA	International Conference on Object-Oriented Programming, Systems, Languages, and Applications	1
ESEC/FSE	International Symposium on the Foundations of Software Engineering / European Software Engineering Conference	1
TSE	IEEE Transaction on Software Engineering	1
IST	Information and Software Technology	3
SCP	Science of Computer Programming	2
ESE	Empirical Software Engineering	4

III. Survey Process

The survey process started by searching through 11 conference proceedings and 4 journals in the SE field[1]. All the conferences and journals are listed in Table 1. To search the source, the search keywords used were "topic model" and "software". During this process, the "title", "keyword", or "abstract" field is used for indexing. As the LDA topic model was originated from 2003, we restricted the search scope between 2003 and 2015.

After the searching process, we need to filter the search results to obtain the relevant results that use topic model for

[1]We also searched other SE conferences and journals, such as ESEM, ISSTA, JSS, TPLAS, TOSEM, SPE and ASE journal, but we did not get any relevant results after the searching and filtering process. So these 15 selected venues are the conferences and journals that have the results we need.

Table II
FILTERING CRITERIA

Inclusion criteria	Exclusion criteria
The study focused on a SE task.	The research area in the study did not belong to SE.
The used technique for the SE task is a topic model.	The study did not consider topic model as a main technique to support the SE task.
The study provided empirical validation of the SE task(s).	The study was just a summarization of discussing topic models used in the SE area.

Table III
THE USED TOPIC MODELS IN SE

Topic model	Count	Reference
Latent Dirichlet Allocation (LDA)	29	[1, 4–6, 14, 20, 21, 23, 24, 28, 29, 36, 38, 41–43, 48–51, 54, 58–63, 66, 68]
Dynamic Topic Model	1	[32]
On-line LDA	1	[31]
Relational topic model	3	[8, 22, 47]
PAM	2	[15, 19]
Citation influence model	1	[33]
Author-Topic Mode	1	[39]

SE tasks. We set the inclusion and exclusion criteria to filter the search results, as shown in Table 2. For a study included in our survey, the study should meet all these three inclusion criteria. For the exclusion criteria, the study is filtered out when only one exclusion criterion is met.

IV. SURVEY RESULTS

Based on the searching and filtering process, we retrieved 38 studies. In terms of the publication venue, 28 papers are from the conference proceedings and 10 papers from the journals. In Figure 1, we show the distribution of the selected papers in different years. The results show that topic models were used in the SE field from 2007 and an increasing number of studies emerge in recent years. For example, there were 11 papers focusing on this area in 2013.

Topic models have been widely used in the SE tasks. The used topic models are either the LDA topic model or its variants. Table 3 shows the statistics of different topics used in SE tasks, which shows that LDA is the most widely used topic model, followed by the relational topic model. For other topic models, there is still a few studies. At present, topic models are applied in various SE tasks, including source code comprehension, feature location, software defects prediction, developer recommendation, traceability link recovery, refactoring, software history comprehension, software testing and social software engineering. These tasks are performed either on the source code or other documents such as requirement documentation, bug reports, commits and so on. Table 4 summarizes the topic models used in various SE tasks. We notice that that there are respectively 12 studies focusing on exploring topic models in comprehension of source code and software historical repositories. In addition, there are seven studies which used topic models for software defects prediction in recent years. For other SE tasks, such as feature location, refactoring, regression testing, there is still a few studies on them.

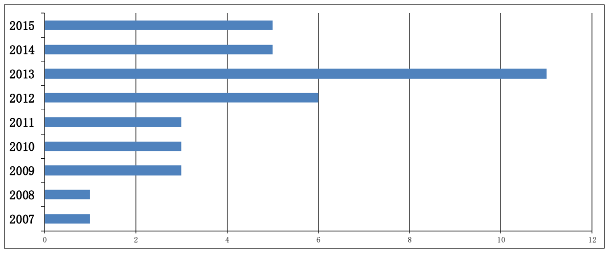

Figure 1. The distribution of the papers published in different years

Table IV
TOPIC MODELS USED IN THE SE TASKS

SE tasks	Count	Reference
Source code comprehension	12	[5, 14, 15, 19, 22–24, 29, 42, 47, 62, 68]
Feature location	1	[54]
Software defects prediction	7	[33, 36, 43, 49–51, 59]
Developer recommendation	1	[39]
Traceability link recovery	3	[28, 54, 59]
Refactoring	1	[8]
Software history comprehension	12	[4, 20, 21, 31, 32, 38, 41, 48, 58, 60, 66, 68]
Software regression testing	1	[61]
Social software engineering	3	[1, 6, 63]

V. TOPIC MODELS IN SOFTWARE ENGINEERING

A. Source Code Comprehension

Program comprehension is the first task when dealing with the code at hand. Clustering is a commonly used technique with topic models in code decomposition to enhance program understanding or reduce a developer's searching effort [34].

Kuhn et al. proposed an approach to consistent layout for program visualization, Software Cartography, in which the position of a software artifact reflects its vocabulary, and distance corresponds to similarity of vocabulary [35]. They use LSI to map software artifacts to a vector space, and then use Multidimensional Scaling to map this vector space down to a two-dimension display for program comprehension. Liu et al. proposed a novel program feature network to identify the semantic features of the program at class level [40]. PFN is generated based on the relational topic model and it can be used to predict the possible links between the newly change request in existing program feature network. Savage et al. proposed an approach, TopicXP, to help developers gain an overview of a software system under analysis by extracting and visualizing natural language topics [57]. It is performed by extracting and analyzing unstructured information in source code identifier names and comments using LDA.

B. Feature Location

The task of concept location (or feature location) is to identify the parts (e.g., documents or methods) of the source code that implement a given feature of the software system [17]. It is usually used to identify an initial location in the source code that implements a change request in a software system.

Poshyvanyk et al. combined Formal Concept Analysis and LSI together for feature location [55]. LSI is used to map concepts expressed in textual change requests to relevant parts of the source code. The search results are then organized in a concept lattice using Formal Concept Analysis. Revelle et al. performed data fusion between LSI, dynamic analysis, and web mining algorithms to perform feature location [18]. The basis of the approach is that combining information from multiple sources is more effective than using the information individually. Nie et al. proposed an approach to locate the interested source code based on LDA [52]. To improve the effects of feature location, the approach uses the topic cohesion and coupling based on a software dependency network.

C. Software Defects Prediction

A software defect is an error, bug, mistake, failure, or fault in a program or system that generates an unexpected outcome, or precludes the software from behaving as intended. The results of defect prediction methods produce a list of defect-prone source code artifacts so that quality assurance teams can effectively allocate appropriate resources on the defect-prone source code.

Nguyen et al. proposed an approach to use topic modeling to measure the concerns in source code [51]. They use them as the input for machine learning-based defect prediction models to predict software defects. Andrzejewski et al. proposed a novel Delta-Latent-Dirichlet-Allocation model, which models execution traces attributed to failed runs of a program as being generated by two types of latent topics: normal usage topics and bug topics [3]. Chen et al. proposed to use a statistical topic modeling technique to approximate software concerns as topics [14]. Then, various metrics on these topics are used to help explain the defect-proneness of the source code. Hu et al. leveraged the relation strength information to help predict defect-prone source code [33]. The relation strength is measured by a citation influence topic model to determine dependency strengths among source code elements and developers.

D. Developer Recommendation

In the process of software maintenance, software changes need to be implemented effectively and efficiently. In order to improve the effectiveness and the quality of software maintenance, assigning issue request to the developer(s) who is/are the most suitable is necessary.

Xie et al. proposed an approach, DRETOM, which recommends appropriate developers for bug resolution in collaborative behavior [64]. DRETOM models developers' development experience and expertise on bug resolving activities based on topic models that are built from their historical bug resolving records. Linstead et al. applied the statistical author-topic models for developer recommendation [39]. Rigor's approach mines developers' contributions and competencies from a given code base while simultaneously extracting software function in the form of topics. Zhang et al. proposed to combine topic model and developer relations (e.g., bug reporter and assignee) to capture developers' interest and experience on specific bug reports [67]. Yang et al. proposed an approach to recommend the most appropriate developer to fix each bug and predict its severity [65]. When a new bug report arrives, they identify corresponding reports that have similar topic(s) to which the report belongs for recommendation.

E. Traceability Link Recovery

Traceability link recovery seeks to connect different types of software artifacts (e.g., documentation with source code) in different software repositories [27]. Traceability link recovery aims to automatically uncover links between pairs of software artifacts, such as source code and requirements documents.

McMillan et al. used both textual (via LSI) and structural (via Evolving Inter-operation Graphs) information to recover traceability links between source code and requirements documents [44]. Oliveto et al. conducted an empirical study to statistically analyze the equivalence of several traceability link recovery methods based on Information Retrieval techniques (including topic models) [53]. The results show that VSM and LSI are almost equivalent, while LDA is able to capture a dimension unique to the other techniques. Borg et al. conducted a map of work on information retrieval based traceability link recovery studies [12]. Hence, more details of using topic models for traceability link recovery can refer to their work [12].

F. Refactoring

During software maintenance and evolution, the structure of the software system undergoes continuous changes, which may drift the source code away from its original design. Thus the quality of the evolving software may be deteriorated. Software refactoring is a process that can overcome this problem and reduce the costs associated with software evolution. Software refactoring is used as an internal modification of source code to improve system quality without modifying the system's behavior.

Bavota et al. proposed an approach, Methodbook, which utilizes relational topic models to analyze both structural and textual information to support move method refactoring [8]. In addition, they also used relational topic models to analyze underlying latent topics in source code as well as structural dependencies to recommend refactoring operations, which aims at moving a class to a more suitable package [7].

G. Software History Comprehension

During software evolution, software developers and stakeholders often seek ways to know the history and evolution of certain aspects of the system to help understand software development. Both how a software system changes (e.g., it grows rapidly every twelfth month) and why a software system changes (e.g., bug fix, new feature) can shed light on the processes of the development iteration. Such kind of work is usually been done by retrieving software histories (e.g., revision control systems, bug tracking systems).

Thomas et al. applied LDA to proposed a Hall Model [26] to analyze the history of source code documents to discover and monitor the drift of topics in source code [60]. They applied LDA to all of the versions of the system at once, then mapped the topics back to individual version of the system. Hindle et al. applied LDA to propose a Link Model [30] to mine the history of commit messages to recover the development topics during a time interval [45]. They applied LDA to each time window of the corpus separately, then linked topics in different time windows together according to a similarity measure. Hu et al. applied Dynamic Topic Models and on-line model on the commit messages within revision control systems to explore the evolution of development topics (e.g., bug fix, module remove) over time [31, 32]. In their approach, topic strength and content evolution can be extracted simultaneously, which means we can not only understand the evolution of the strength of a topic, but also the evolution of different aspects within.

H. Software Regression Testing

Software regression testing is one of the most practical means to ensure software quality during program evolution. As software constantly evolves, the test suite tends to grow. Thus it is very costly to execute the original entire test suite. Hence, researchers and practitioners seek to reduce the effort required for regression testing in various ways, such as test case selection and test case prioritization.

Thomas et al. applied the topic model to static test case prioritization [61]. In their approach, topic model is performed on the linguistic data of test cases to approximate their functionality, and high priority is given to test cases that test different functionalities of the system under test. Their approach focuses mainly on black-box test case prioritization.

VI. Conclusions

In this paper, a survey of exploring topic models in SE tasks is presented based on 38 selected publications from 2003 to 2015. From the survey, we see that topic models are widely used in various SE tasks in an increasing tendency. These tasks include source code comprehension, feature location, traceability link recovery, refactoring, software testing, developer recommendation, software defects prediction, software history comprehension, and social software engineering. Among the topic models, LDA is the most widely used topic model. Among different SE tasks, source code comprehension and software history comprehension are the most.

VII. Acknowledgments

This work is supported partially by Natural Science Foundation of China under Grant No. 61402396 and No. 61472344, partially by the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant No. 13KJB520027, and partially by the Research Innovation Program for College Graduates of Jiangsu Province under Grant No. SJLX15_0665.

References

[1] M. Allamanis and C. Sutton. Why, when, and what: Analyzing stack overflow questions by topic, type, and code. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 53–56, 2013.

[2] L. AlSumait, D. Barbará, and C. Domeniconi. On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *Data Mining, 2008. Eighth IEEE International Conference on*, pages 3–12. IEEE, 2008.

[3] D. Andrzejewski, A. Mulhern, B. Liblit, and X. Zhu. Statistical debugging using latent topic models. In *Proceedings of the 18th European Conference on Machine Learning*, pages 6–17, 2007.

[4] S. K. Bajracharya and C. V. Lopes. Mining search topics from a code search engine usage log. In *Proceedings of the 6th International Working Conference on Mining Software Repositories*, pages 111–120, 2009.

[5] P. Baldi, C. V. Lopes, E. Linstead, and S. K. Bajracharya. A theory of aspects as latent topics. In *Proceedings of the 23rd Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 543–562, 2008.

[6] A. Barua, S. W. Thomas, and A. E. Hassan. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Softw. Engg.*, 19(3):619–654, June 2014.

[7] G. Bavota, M. Gethers, R. Oliveto, D. Poshyvanyk, and A. D. Lucia. Improving software modularization via automated analysis of latent topics and dependencies. *ACM Trans. Softw. Eng. Methodol.*, 23(1):4, 2014.

[8] G. Bavota, R. Oliveto, M. Gethers, D. Poshyvanyk, and A. D. Lucia. Methodbook: Recommending move method refactorings via relational topic models. *IEEE Trans. Software Eng.*, 40(7):671–694, 2014.

[9] D. M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, 2012.

[10] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *Machine Learning, Proceedings of the Twenty-Third International Conference*, pages 113–120, 2006.

[11] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[12] M. Borg, P. Runeson, and A. Ardö. Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability. *Empirical Software Engineering*, 19(6):1565–1616, 2014.

[13] J. Chang and D. M. Blei. Relational topic models for document networks. In D. A. V. Dyk and M. Welling, editors, *AISTATS*, volume 5 of *JMLR Proceedings*, pages 81–88. JMLR.org, 2009.

[14] T.-H. Chen, S. W. Thomas, M. Nagappan, and A. E. Hassan. Explaining software defects using topic models. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, pages 189–198, 2012.

[15] C. S. Corley, E. A. Kammer, and N. A. Kraft. Modeling the ownership of source code topics. In *IEEE 20th International Conference on Program Comprehension*, pages 173–182, 2012.

[16] L. Dietz, S. Bickel, and T. Scheffer. Unsupervised prediction of citation influences. In *Machine Learning,*

Proceedings of the Twenty-Fourth International Conference, pages 233–240, 2007.

[17] B. Dit, M. Revelle, M. Gethers, and D. Poshyvanyk. Feature location in source code: a taxonomy and survey. *Journal of Software: Evolution and Process*, 25(1):53–95, 2013.

[18] B. Dit, M. Revelle, and D. Poshyvanyk. Integrating information retrieval, execution and link analysis algorithms to improve feature location in software. *Empirical Software Engineering*, 18(2):277–309, 2013.

[19] B. P. Eddy, J. A. Robinson, N. A. Kraft, and J. C. Carver. Evaluating source code summarization techniques: Replication and expansion. In *International Conference on Program Comprehension*, pages 13–22, 2013.

[20] Y. Fu, M. Yan, X. Zhang, L. Xu, D. Yang, and J. D. Kymer. Automated classification of software change messages by semi-supervised latent dirichlet allocation. *Information & Software Technology*, 57:369–377, 2015.

[21] L. V. Galvis Carreño and K. Winbladh. Analysis of user comments: An approach for software requirements evolution. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 582–591, Piscataway, NJ, USA, 2013. IEEE Press.

[22] M. Gethers and D. Poshyvanyk. Using relational topic models to capture coupling among classes in object-oriented software systems. In *26th IEEE International Conference on Software Maintenance*, pages 1–10, 2010.

[23] S. Grant and J. R. Cordy. Examining the relationship between topic model similarity and software maintenance. In *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering*, pages 303–307, 2014.

[24] S. Grant, J. R. Cordy, and D. B. Skillicorn. Using heuristics to estimate an appropriate number of latent topics in source code analysis. *Sci. Comput. Program.*, 78(9):1663–1678, 2013.

[25] S. Grimes. Unstructured data and the 80 percent rule. *Carabridge Bridgepoints*, 2008.

[26] D. Hall, D. Jurafsky, and C. D. Manning. Studying the history of ideas using topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 363–371, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[27] A. E. Hassan. The road ahead for mining software repositories. 2008.

[28] A. Hindle, C. Bird, T. Zimmermann, and N. Nagappan. Do topics make sense to managers and developers? *Empirical Software Engineering*, December 2015. To appear.

[29] A. Hindle, N. A. Ernst, M. W. Godfrey, and J. Mylopoulos. Automated topic naming to support cross-project analysis of software maintenance activities. In *Proceedings of the 8th International Working Conference on Mining Software Repositories*, pages 163–172, 2011.

[30] A. Hindle, M. W. Godfrey, and R. C. Holt. What's hot and what's not: Windowed developer topic analysis. In *ICSM*, pages 339–348. IEEE, 2009.

[31] J. Hu, X. Sun, and B. Li. Explore the evolution of development topics via on-line lda. In *Proceedings of the 22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering*, SANER '15, 2015.

[32] J. Hu, X. Sun, D. Lo, and B. Li. Modeling the evolution of development topics using dynamic topic models. In *Proceedings of the International Conference on Software Analysis, Evolution, and Reengineering*, 2015.

[33] W. Hu and K. Wong. Using citation influence to predict software defects. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 419–428, Piscataway, NJ, USA, 2013. IEEE Press.

[34] A. Kuhn, S. Ducasse, and T. Girba. Enriching reverse engineering with semantic clustering. In *Proceedings of the 12th Working Conference on Reverse Engineering*, pages 133–142, 2005.

[35] A. Kuhn, D. Erni, P. Loretan, and O. Nierstrasz. Software cartography: Thematic software visualization with consistent layout. *J. Softw. Maint. Evol.*, 22(3):191–210, Apr. 2010.

[36] T. B. Le, F. Thung, and D. Lo. Predicting effectiveness of ir-based bug localization techniques. In *25th IEEE International Symposium on Software Reliability Engineering*, pages 335–345, 2014.

[37] W. Li and A. McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Machine Learning, Proceedings of the Twenty-Third International Conference*, pages 577–584, 2006.

[38] E. Linstead and P. Baldi. Mining the coherence of GNOME bug reports with statistical topic models. In *Proceedings of the 6th International Working Conference on Mining Software Repositories*, pages 99–102, 2009.

[39] E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi. Mining eclipse developer contributions via author-topic models. In *Proceedings of the Fourth International Workshop on Mining Software Repositories*, MSR '07, pages 30–, Washington, DC, USA, 2007. IEEE Computer Society.

[40] X. Liu, X. Sun, B. Li, and J. Zhu. PFN: A novel program feature network for program comprehension. In *2014 IEEE/ACIS 13th International Conference on Computer and Information Science*, pages 349–354, 2014.

[41] N. Lopez. Using topic models to understand the evolution of a software ecosystem. In *Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 723–726, 2013.

[42] A. D. Lucia, M. D. Penta, R. Oliveto, A. Panichella, and S. Panichella. Using ir methods for labeling source code artifacts: Is it worthwhile? In D. Beyer, A. van Deursen, and M. W. Godfrey, editors, *ICPC*, pages 193–202, 2012.

[43] S. K. Lukins, N. A. Kraft, and L. H. Etzkorn. Bug localization using latent dirichlet allocation. *Information & Software Technology*, 52(9):972–990, 2010.

[44] C. McMillan, D. Poshyvanyk, and M. Revelle. Combining textual and structural analysis of software artifacts for

traceability link recovery. In *ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, pages 41–48, 2009.

[45] Q. Mei and C. Zhai. Discovering evolutionary theme patterns from text: An exploration of temporal text mining. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 198–207, New York, NY, USA, 2005. ACM.

[46] T. Mens, A. Serebrenik, and A. Cleve, editors. *Evolving Software Systems*. Springer, 2014.

[47] E. Moritz, M. L. Vásquez, D. Poshyvanyk, M. Grechanik, C. McMillan, and M. Gethers. Export: Detecting and visualizing API usages in large source code repositories. In *2013 28th IEEE/ACM International Conference on Automated Software Engineering*, pages 646–651, 2013.

[48] S. Neuhaus and T. Zimmermann. Security trend analysis with CVE topic models. In *IEEE 21st International Symposium on Software Reliability Engineering*, pages 111–120, 2010.

[49] A. T. Nguyen, T. T. Nguyen, J. M. Al-Kofahi, H. V. Nguyen, and T. N. Nguyen. A topic-based approach for narrowing the search space of buggy files from a bug report. In *26th IEEE/ACM International Conference on Automated Software Engineering*, pages 263–272, 2011.

[50] A. T. Nguyen, T. T. Nguyen, T. N. Nguyen, D. Lo, and C. Sun. Duplicate bug report detection with a combination of information retrieval and topic modeling. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, pages 70–79, New York, NY, USA, 2012. ACM.

[51] T. T. Nguyen, T. N. Nguyen, and T. M. Phuong. Topic-based defect prediction (nier track). In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 932–935, New York, NY, USA, 2011. ACM.

[52] K. Nie and L. Zhang. Software feature location based on topic models. In K. R. P. H. Leung and P. Muenchaisri, editors, *APSEC*, pages 547–552. IEEE, 2012.

[53] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia. On the equivalence of information retrieval methods for automated traceability link recovery. In *Proceedings of the 2010 IEEE 18th International Conference on Program Comprehension*, ICPC '10, pages 68–71, Washington, DC, USA, 2010. IEEE Computer Society.

[54] A. Panichella, B. Dit, R. Oliveto, M. D. Penta, D. Poshyvanyk, and A. D. Lucia. How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In *35th International Conference on Software Engineering*, pages 522–531, 2013.

[55] D. Poshyvanyk, M. Gethers, and A. Marcus. Concept location using formal concept analysis and information retrieval. *ACM Trans. Softw. Eng. Methodol.*, 21(4):23, 2012.

[56] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pages 487–494, Arlington, Virginia, United States, 2004. AUAI Press.

[57] T. Savage, B. Dit, M. Gethers, and D. Poshyvanyk. Topicxp: Exploring topics in source code using latent dirichlet allocation. In *26th IEEE International Conference on Software Maintenance*, pages 1–6, 2010.

[58] X. Sun, B. Li, H. Leung, B. Li, and Y. Li. Msr4sm: Using topic models to effectively mining software repositories for software maintenance tasks. *Information and Software Technology*, 66(0):1 – 12, 2015.

[59] S. W. Thomas. Mining software repositories using topic models. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 1138–1139, 2011.

[60] S. W. Thomas, B. Adams, A. E. Hassan, and D. Blostein. Studying software evolution using topic models. *Sci. Comput. Program.*, 80:457–479, Feb. 2014.

[61] S. W. Thomas, H. Hemmati, A. E. Hassan, and D. Blostein. Static test case prioritization using topic models. *Empirical Softw. Engg.*, 19(1):182–212, Feb. 2014.

[62] K. Tian, M. Revelle, and D. Poshyvanyk. Using latent dirichlet allocation for automatic categorization of software. In *Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories*, MSR '09, pages 163–166, Washington, DC, USA, 2009. IEEE Computer Society.

[63] M. L. Vásquez, B. Dit, and D. Poshyvanyk. An exploratory analysis of mobile development issues using stack overflow. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 93–96, 2013.

[64] X. Xie, W. Zhang, Y. Yang, and Q. Wang. Dretom: Developer recommendation based on topic models for bug resolution. In *Proceedings of the 8th International Conference on Predictive Models in Software Engineering*, PROMISE '12, pages 19–28, New York, NY, USA, 2012. ACM.

[65] G. Yang, T. Zhang, and B. Lee. Towards semi-automatic bug triage and severity prediction based on topic model and multi-feature of bug reports. In *Proceedings of the 2014 IEEE 38th Annual Computer Software and Applications Conference*, COMPSAC '14, pages 97–106, Washington, DC, USA, 2014. IEEE Computer Society.

[66] S. Yu. Retrieving software maintenance history with topic models. In *28th IEEE International Conference on Software Maintenance*, pages 621–624, 2012.

[67] T. Zhang, G. Yang, B. Lee, and E. K. Lu. A novel developer ranking algorithm for automatic bug triage using topic model and developer relations. In *Proceedings of the 21st Asia-Pacific Software Engineering Conference*, APSEC '14, 2014.

[68] C. Zou and D. Hou. LDA analyzer: A tool for exploring topic models. In *30th IEEE International Conference on Software Maintenance and Evolution*, pages 593–596, 2014.