Mining Textual Data for Software Engineering Tasks

Latifa Guerrouj
McGill University
3661 Peel St., Canada H3A 1X1
Mobile: (+1) 514-791-0085
Email: latifa.guerrouj@polymtl.ca
Web: http://latifaguerrouj.ca/

Benjamin C. M. Fung
McGill University
3661 Peel St., Canada H3A 1X1
Phone: (+1) 514-398-3360
Fax: (+1) 514-398-7193
Email: ben.fung@mcgill.ca
Web: http://dmas.lab.mcgill.ca/fung/index.htm

David Lo
Singapore Management University
80 Stamford Road
Singapore 178902
Email: davidlo@smu.edu.sg
Web: http://www.mysmu.edu/faculty/davidlo/

Foutse Khomh
École Polytechnique de Montréal
2500, chemin de la Polytechnique, Montral (Qubec) H3T 1J4
Phone: (+1) 514-340-4711
Fax: (+1) 514-340-5139
Email: foutse.khomh@polymtl.ca
Web: http://khomh.net/

Abdelwahab Hamou-Lhadj
Concordia University
515 St. Catherine, West
Montréal, H3G 2W1 Canada
Phone: (+1) 514-848-2424 ext 7949
Email: wahab.hamou-lhadj@concordia.ca
Web: http://users.encs.concordia.ca/ abdelw/index.html

***Abstract*—Software development artifacts produced during the development process are of different types. Some are structured such as the source code and execution traces while others are unstructured like source code comments, identifiers, bug reports, usage logs, etc. Such data embeds a significant knowledge about software projects that can help software developers make technical and business decisions.**

While the focus has been extensively on source code in the past, researchers have recently investigated the textual information (*e.g.*, identifiers and comments) contained in software artifacts or informal documentation (*e.g.*, StackOverflow, emails threads, change logs, bug reports, etc.) about the software systems. Automatic techniques and tools have been developed to generate and–or mine unstructured data to gain insight about the software development process or assist development teams in tasks like software traceability, feature/concept location, source code vocabulary normalization, bug localization, and summarization.

The tutorial will start with an introduction of textual information in source code and–or documentation. Next, we will present automatic techniques and tools to generate and mine unstructured data and discuss related challenges. We will also present examples of major software engineering tasks making use of unstructured data mining along with scenarios of their application and the most recent contributions relevant to each task. Specifically, we will focus on automatic source code vocabulary normalization, summarization, crash reports analysis for fault localisation. Finally, we will discuss with the audience the success and failures in achieving the full potential of such tasks in a software development context as well as possible improvements and research directions. The tutorial will provide novice with a common framework about major software engineering tasks leveraging textual information while for experts, the tutorial can be an interesting opportunity to discuss challenges, document the state of the art and practice, encourage cross-fertilization across various research areas ranging from mining software repositories to natural language processing and text retrieval, and to establish foreseeable collaborations between researchers.

I. Motivation

Software development projects knowledge is grounded in rich data. For example, source code, check-ins, bug reports, work items and test executions are recorded in software repositories such as version control systems (Git, Subversion, Mercurial, CVS) and issue-tracking systems (Bugzilla, JIRA, Trac), and the information about user experiences of interacting with software is typically stored in log files or informal documentation such as StackOverflow.

While there has been extensive research on static analysis of source code, recent studies have exploited textual information used in source code of software systems or trapped in informal documentation (*e.g.*, emails threads, StackOverflow posts, etc.). The purpose is to develop automatic software engineering techniques, gain insights and understand software projects, and support the decision-making process.

Major software engineering tasks have leveraged textual information. For example in the context of software traceability, researchers have made use of textual information to trace code to documents (*e.g.*, requirements) [1], [2], they also suggested lightweight techniques of linking code to documentation such as email threads [3] and StackOverflow [4], as well as tracing code examples to documentation [5]. Textual information have been also exploited in feature/concept location [6], [7], [8], source code vocabulary normalization [9], [10] and summarization of complex artifacts involving release notes [11], StackOverflow [12], and bug reports [13]. Such approaches have been developed with the aim of guiding developers and practitioners towards a better understanding of their software projects and the way they evolve.

While solutions provided for these engineering tasks demonstrated promising results, there are many challenges left concerned with mining textual information, using it in the development of the above-mentioned tasks, as well as integrating and adopting such solutions into software development processes.

The goals of this tutorial are to discuss the use of textual information, its related challenges and open-question, tools and techniques of mining such data as well as ways of integrating and exploiting them by major software engineering tasks to fully reap their benefits.

We invite both novice and experts to this tutorial that will be an opportunity to share tools, techniques, and experiences in the field. We also plan, after the presentation of the tutorial, to have a discussion and dissemination of the presented research by opening up a discussion and involving participants in sharing their opinions. We invite researchers and practitioners interested in improving, integrating, and adopting the use and mining of textual information in their software engineering tools and thus software development and maintenance activities. The tutorial encourages both academic researchers and industrial practitioners for an exchange of ideas and collaboration.

II. Topics

The tutorial will focus on the presentation of recent techniques and tools used to generate and mine textual information as well as software engineering tasks making use of such rich data.

The tutorial will explain, present, and discuss the following:

1) Textual information in source code and informal documentation;
2) Benefits of using textual information in software engineering tasks;
3) Recent tools and techniques used to generate and mine textual information;
4) Challenges related to mining textual information;
5) Major software engineering tasks using textual information;
6) Explain source code vocabulary normalization and how it makes use of textual information along with recent automatic approaches;
7) Present summarization software artifacts with recent automatic approaches in this area;
8) Explore bug localization, how it makes use of textual information, and how the instructors could improve it by leveraging text in crash reports;
9) Identification of open research challenges and possible solutions.

III. Presenters' Experience in the Area and Topics of Their Presentations

Latifa Guerrouj preformed her past studies on context-aware source code vocabulary normalization. Vocabulary normalization aligns the vocabulary found in the source code with that found in other software artifacts (*e.g.*, test cases, requirements, specifications, design, etc). Latifa developed automatic context-aware source code vocabulary approaches by mining textual information in source code [14], [15], [16], [17], [18]. She also investigated the use of normalization in the context of feature location using textual information and dynamic analysis [19]. Recently, she suggested a new approach summarizing Android API classes and methods discussed in StackOverflow using n-grams language models and applying machine learning techniques [12]. Latifa is the co-organizer of the International Workshop on Software Analytics (SWAN'15). In this tutorial, she will make the focus on how text found in source code or information documentation can be mined and exploited in the context of engineering tasks namely source code vocabulary and summarization of software artefacts.

David Lo research work focuses on software engineering and data mining. He investigates how techniques from these two research areas could benefit and complement each other. In the software engineering area, his research includes software specification mining/protocol inference, mining software repositories, program analysis, software testing and automated debugging. Technique-wise, he investigates a composition of techniques including static analysis, dynamic analysis, data mining, information retrieval, and natural language processing. In the data mining area, his works on frequent pattern mining, discriminative pattern mining, and social network mining. David contributed to the analysis of software text with the aim of aiding software developers in performing their various tasks. Examples of his works relevant to this tutorial involve enhanced techniques making use of text version for bug localization [20], a large scale investigation of issue trackers from GitHub [21], accurate information retrieval-based bug localization based on bug reports [22], interactive fault localization leveraging simple user feedback [23], automatic duplicate bug report detection with a combination of information retrieval and topic modeling [24]. David is the co-organizer of the first International Workshop on Machine Learning and Information Retrieval for Software Evolution (MALIR-SE) collocated with ASE 2013. In this tutorial, David will make the focus on techniques of mining text and its use for bug localization.

Foutse Khomh leads the SoftWare Analytics and Technologies (SWAT) Lab that applies analytic techniques to empower development teams with insightful and actionable information about their activities. SWAT team also build tools to assess and improve the quality of software systems. Early models and tools proposed by SWAT members are already being used in the industry. Among Foutse's research works related to this workshop, we state the ones on challenges and issues of mining crash reports [25], tracking back the history of commits in low-tech reviewing environments [26], supplementary bug fixes vs. re-opened bugs [27], improving bug localization using correlations in crash reports [28], classifying field crash reports for fixing bugs: A case study of Mozilla Firefox [29], and a text-based approach to classify change requests [30]. Foutse co-founded the International Workshop on Release

Engineering (RELENG) in 2013 and has been co-organizing it since then. In this tutorial, Foutse will show his recent work on using crash reports for the improvement of bug localization and identifying highly impactful bugs.

IV. Goals and Expected Results

This tutorial targets both novice and experts working in the field of software maintenance and evolution, interested in the analysis of software text, its mining, and its practical use in the context of software engineering tasks. For experts, it will provide an informal interactive forum to exchange ideas and experiences, streamline research making use of textual information, identify some common ground of their work, and share lessons and challenges, thereby articulating a vision for the future of software engineering.

The intended outcomes of this tutorial are:

1) Make clear (for novice) what is textual information and techniques of its mining;
2) Explore the different contemporary software engineering techniques making use of textual data;
3) Stimulate discussions, interest, and understanding in integrating textual info in software engineering tasks and software development process;
4) Bridging the gap between the theory and practice by bringing together researchers and practitioners interested in analysing software text for software engineering tasks;
5) Discuss challenges, experiences, lessons, and explore the different possible strategies to overcome the challenges faced and towards promising solutions to essential problems;
6) Build a common framework of major automatic approaches making use of textual information;
7) Advance the state of the art and practice in software engineering;

V. Outline

1) Introduction about software text and tools to generate and mine such data by David Lo.
2) Exploration of major software engineering tasks making use of textual data by Foutse Khomh.
3) Presentation of source code vocabulary normalization along with examples of recent published automatic source code vocabulary normalization approaches by Latifa Guerrouj.
4) Presentation of summarization of software artifacts along with examples of recent published automatic summarization approach by Latifa Guerrouj.
5) Presentation of bug localization with examples of most recent automatic approaches in this area by David Lo.
6) Exploration of recent ways to improve bug localization using crash reports and to identify impactful bugs by Foutse Khomh.
7) Summary and recap of the tutorial by David Lo, Latifa Guerrouj, and Foutse Khomh.

VI. Target audience

This tutorial is intended for both novice and experts, academics and industrial practitioners. It will provide participants with an understanding of software text, techniques to mine it from source code or documentation, and ways of adopting and integrating it in major engineering tasks. Additionally, novice will be able to understand engineering tasks such as vocabulary normalization, bug localization, and summarization and how they exploit textual data to fully reap their benefits. The tutorial will show scenarios of the presented approaches and how they can help to guide developers during their tasks as well as to improve software maintenance and evolution.

We will also discuss the limitations and challenges of the most recent related techniques and how these issues can be addressed and mitigated.

Participants are encouraged to talk about their recent works related to the tutorial (if any) and share their experiences and major faced challenges. Experts will be there to guide and provide them with feedback.

VII. Format

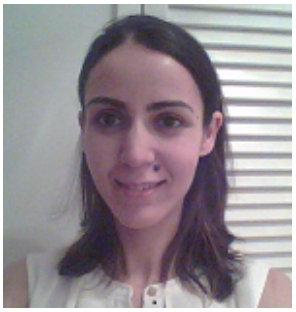
We propose to have 2-hours tutorial consisting of a 1 hour dedicated to an 1) introduction of textual data by the presenters, 2) major software engineering tasks leveraging such data, 3) concrete examples of recent automatic approaches on source code vocabulary normalization and summarization, and 4) related discussions by participants. The other 1 hour will be devoted to the 5) bug localization, 6) its enhancement using crash reports as well as ways of identifying impactful bugs, 7) discussion by participants, and 8) summary and recap.

We encourage discussions so as to develop an in-depth understanding of the presented topics for novice. Experts are invited to enrich the discussions by providing opinions and moderating a discussion on the state-of-the-art and state-of-the-practice of software engineering tasks making use of textual data.

VIII. Acknowledgement

Special thanks to Giuliano Antoniol and Massimiliano Di Penta for all their valuable feedback on this tutorial.

IX. Contributors' biography



Latifa Guerrouj is a Postdoctoral Research Fellow at McGill University, Canada. She received her Ph.D. from the Department of Computing and Software Engineering (DGIGL) of École Polytechnique de Montréal, Canada. Her research work/interests involves empirical software engineering, software analytics, data mining, and big data software engineering. Latifa is serving as an organizing and program committee member for several international conferences and workshops including ICSME'16, ICSME'15, SANER'15, SWAN'15, ICSM'14, SCAM'14, MSR'14/13, WCRE'13/12, ICST'12, and MUD'12/13. She is a member of ACM and IEEE.

Benjamin C. M. Fung is an Associate Professor of Information Studies (SIS) at McGill University and a Research Scientist in the National Cyber-Forensics and Training Alliance Canada (NCFTA Canada). He received a Ph.D. degree in computing science from Simon Fraser University in 2007. Dr. Fung has over 80 refereed publications that span the prestigious research forums of data mining, privacy protection, cyber forensics, services computing, and building engineering. His data mining works in crime investigation and authorship analysis have been reported by media worldwide. His research has been supported in part by the Discovery Grants and Strategic Project Grants from the Natural Sciences and Engineering Research Council of Canada (NSERC), Insight Development Grants from the Social Sciences and Humanities Research Council (SSHRC), Defence Research and Development Canada (DRDC), and Fonds de recherche du Qubec - Nature et technologies (FRQNT), and NCFTA Canada. Dr. Fung is a licensed professional engineer in software engineering, and is currently affiliated with the Data Mining and Security Lab at SIS.

David Lo is an Assistant Professor in the School of Information Systems at Singapore Management University. He received his PhD from School of Computing, National University of Singapore in 2008. Before that, he was studying at School of Computer Engineering, Nanyang Technological University and graduated with a B.Eng (Hons I) in 2004. David works in the intersection of software engineering and data mining. His research interests include dynamic program analysis, specification mining, and pattern mining. Lo received a PhD in computer science from the National University of Singapore. He is a member of the IEEE and the ACM.

Foutse khomh is an Assistant Professor at the École Polytechnique de Montréal, where he heads the SWAT Lab on software analytics and cloud engineering research (http://swat.polymtl.ca/). Prior to this position he was a Research Fellow at Queen's University (Canada), working with the Software Reengineering Research Group and the NSERC/RIM Industrial Research Chair in Software Engineering of Ultra Large Scale Systems. He received his Ph.D in Software Engineering from the University of Montreal in 2010, under the supervision of Yann-Gaël Guéhéneuc. His main research interest is in the field of empirical software engineering, with an emphasis on developing techniques and tools to improve software quality. Over the years, he has applied many text mining techniques to solve multiple software engineering problems. He co-founded the International Workshop on Release Engineering (http://releng.polymtl.ca) and was one of the editors of the first special issue on Release Engineering in the IEEE Software magazine.

Abdelwahab Hamou-Lhadj is a tenured Associate Professor in ECE, Concordia University. His research interests include software modeling, software behavior analysis, software maintenance and evolution, anomaly detection systems. He holds a Ph.D. degree in Computer Science from the University of Ottawa (2005). He is a Licensed Professional Engineer in Quebec, and a long- lasting member of IEEE and ACM.

REFERENCES

[1] N. Ali, Y.-G. Guéhéneuc, and G. Antoniol, "Trustrace: Mining software repositories to improve the accuracy of requirement traceability links," *IEEE Transactions on Software Engineering*, vol. 39, no. 5, pp. 725–741, 2013.

[2] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, "Recovering traceability links between code and documentation," *IEEE Transactions on Software Engineering*, vol. 28, no. 10, pp. 970–983, 2002.

[3] A. Bacchelli, M. Lanza, and R. Robbes, "Linking e-mails and source code artifacts," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, 2010, pp. 375–384.

[4] P. C. Rigby and M. P. Robillard, "Discovering essential code elements in informal documentation," in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE '13, 2013, pp. 832–841.

[5] S. Subramanian, L. Inozemtseva, and R. Holmes, "Live api documentation," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014, 2014, pp. 643–652.

[6] D. Liu, A. Marcus, D. Poshyvanyk, and V. Rajlich, "Feature location via information retrieval based filtering of a single scenario execution trace." in *ASE'07*, 2007, pp. 234–243.

[7] D. Poshyvanyk, Y.-G. Guéhéneuc, A. Marcus, G. Antoniol, and V. Rajlich, "Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval," *IEEE Transactions on Software Engineering*, vol. 33, no. 6, pp. 420–432, 2007.

[8] T. Eisenbarth, R. Koschke, and D. Simon, "Locating features in source code," *IEEE Transactions on Software Engieering*, pp. 210–224, March 2003.

[9] L. Guerrouj, D. P. Massimiliano, G. Yann-Gaël, and G. Antoniol, "Tidier: an identifier splitting approach using speech recognition techniques," *Journal of Software: Evolution and Process*, pp. 575–599, 2013.

[10] E. Enslen, E. Hill, L. L. Pollock, and K. Vijay-Shanker, "Mining source code to automatically split identifiers for software analysis," in *Proceedings of of the 6th International Working Conference on Mining Software Repositories*, 2009, pp. 71–80.

[11] L. Moreno, G. Bavota, M. D. Penta, R. Oliveto, and A. Marcus, "How can i use this method," in *Proceedings of the 37th International Conference on Software Engineering*, ser. ICSE 2015, 2015.

[12] L. Guerrouj, D. Bourque, and P. Rigby, "Leveraging informal documentation to summarize classes and methods in context," in *Proceedings of the 37th International Conference on Software Engineering*, ser. ICSE 2015, 2015.

[13] S. Rastkar, G. C. Murphy, and G. Murray, "Summarizing software artifacts: a case study of bug reports." ACM, 2010, pp. 505–514.

[14] L. Guerrouj, M. D. Penta, Y. Guéhéneuc, and G. Antoniol, "An experimental investigation on the effects of context on source code identifiers splitting and expansion," *Empirical Software Engineering*, vol. 19, no. 6, pp. 1706–1753, 2014.

[15] L. Guerrouj, M. D. Penta, G. Antoniol, and Y. G. Guéhéneuc, "Tidier: An identifier splitting approach using speech recognition techniques," *Journal of Software Maintenance - Research and Practice*, p. 31, 2011.

[16] L. Guerrouj, "Normalizing source code vocabulary to support program comprehension and software quality," in *Proceedings of the 2013 International Conference on Software Engineering*, 2013, pp. 1385–1388.

[17] L. Guerrouj, P. Galinier, Y.-G. Guéhéneuc, G. Antoniol, and M. D. Penta, "Tris: a fast and accurate identifiers splitting and expansion algorithm," in *Proc. of the International Working Conference on Reverse Engineering (WCRE'12)*, 2012, pp. 103–112.

[18] N. Madani, L. Guerrouj, M. Di Penta, Y.-G. Guéhéneuc, and G. Antoniol, "Recognizing words from source code identifiers using speech recognition techniques," in *Proceedings of the 14th European Conference on Software Maintenance and Reengineering (CSMR 2010), March 15-18 2010, Madrid, Spain*. IEEE CS Press, 2010.

[19] B. Dit, L. Guerrouj, D. Poshyvanyk, and G. Antoniol, "Can better identifier splitting techniques help feature location?" in *Proc. of the International Conference on Program Comprehension (ICPC)*, Kingston, 2011, pp. 11–20.

[20] S. Wang and D. Lo, "Version history, similar report, and structure: Putting them together for improved bug localization," in *Proceedings of the 22Nd International Conference on Program Comprehension*. ACM, 2014, pp. 53–63.

[21] T. F. Bissyand, D. Lo, L. Jiang, L. Rveillre, J. Klein, and Y. L. Traon, "Got issues? who cares about it? a large scale investigation of issue trackers from github." IEEE, 2013, pp. 188–197.

[22] J. Zhou, H. Zhang, and D. Lo, "Where should the bugs be fixed? - more accurate information retrieval-based bug localization based on bug reports," in *Proceedings of the 34th International Conference on Software Engineering*, 2012, pp. 14–24.

[23] L. Gong, D. Lo, L. Jiang, and H. Zhang, "Interactive fault localization leveraging simple user feedback." IEEE Computer Society, 2012, pp. 67–76.

[24] 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*, 2012, pp. 70–79.

[25] L. An and F. Khomh, "Challenges and issues of mining crash reports," in *1st IEEE International Workshop on Software Analytics, SWAN 2015, Montreal, QC, Canada, March 2, 2015*, 2015, pp. 5–8.

[26] Y. Jiang, B. Adams, F. Khomh, and D. M. German, "Tracing back the history of commits in low-tech reviewing environments," in *Proceedings of the 8th International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Torino, Italy, September 2014.

[27] L. An, F. Khomh, and B. Adams, "Supplementary Bug Fixes vs. Re-opened Bugs." IEEE Computer Society, 2014, pp. 205–214.

[28] S. Wang, F. Khomh, and Y. Zou, in *MSR*, pp. 247–256.

[29] T. Dhaliwal, F. Khomh, and Y. Zou, "Classifying field crash reports for fixing bugs: A case study of mozilla firefox." in *ICSM*. IEEE, 2011, pp. 333–342.

[30] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.-G. 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*, 2008, pp. 23:304–23:318.