Automatic Software Summarization

The State of the Art

Laura Moreno
Department of Computer Science
Colorado State University
Fort Collins, CO, USA
lmorenoc@colostate.edu

Andrian Marcus
Department of Computer Science
The University of Texas at Dallas
Richardson, TX, USA
amarcus@utdallas.edu

ABSTRACT

While automatic text summarization has been widely studied for more than fifty years, in software engineering, automatic summarization is an emerging area that shows great potential and poses new and exciting research challenges. This technical briefing provides an introduction to the state of the art and maps future research directions in automatic software summarization. A first version was presented at ICSE'17 and now it is updated and enhanced, based on feedback from the audience.

1 TOPIC DESCRIPTION

We constantly encounter and produce summaries in our everyday lives, for example: article abstracts, headlines, reviews, movie trailers, tables of contents, etc. According to the American National Standards Institute [2], a summary is "a brief and objective representation of a document or an oral presentation" that allows quickly identification of the basic content of the document by readers and access services. The literature on *text summarization* abounds with definitions of what a summary is or what its purpose is, like the one above. These definitions, however, are rather general and mostly relevant to manually produced summaries. When referring to *automatic text summarization*, more specific definitions can be found. For example, Spärck Jones [3] describes this process as "a reductive transformation of source text to summary text through content reduction by selection and/or generalization on what is important in the source," while Mani and Maybury [4] go beyond, and define it as "the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks)." Notice that the source units, condensation method, target audience and purpose of the summary become essential when automatically summarizing text. These factors impact the summarization process and its output. The source units, for instance, have an influence in the information selection and filtering techniques, while issues like redundancy are more likely to affect *multi-document summarization* than *single-document summarization*.

The focus of this technical briefing is on automatic software summarization, an emerging and growing field in software engineering research, inspired by automatic text summarization. Automatic software summarization refers to the process of generating a concise representation of one or more software artifacts that conveys the information needed by a software stakeholder to perform a particular software engineering task. Different from text summarization, the summarization of software artifacts sometimes involves analysis and processing of source code, a data source specific to the software domain.

1.1 Automatic Software Summarization

Software summarization techniques can be analyzed based on: *input factors*, characterizing the input software artifacts(s); *purpose factors*, defining the transformations necessary to obtain the summary; and *output factors*, characterizing the produced summaries. From the input and output point of view, we categorize here software summarization techniques into:

1. *Text-to-text summarization*: This category groups approaches that generate text-based summaries from textual software artifacts, such as, bug reports (e.g., [10]) or user reviews (e.g., [12]). These approaches are the closest to automatic text summarization and rely on its research.
2. *Code-to-text summarization*: The approaches in this category generate text-based summaries from source code artifacts, such as, methods (e.g., [1, 13]), classes (e.g., [1, 5]), test cases (e.g., [8]), code changes (e.g., [7]), etc.
3. *Code-to-code summarization*: This category includes approaches that generate source code based summaries from source code artifacts, such as, code fragments (e.g., [14]), or code usage examples (e.g., [6]).
4. *Mixed artifact summarization*: This category consists of approaches that generate summaries for heterogeneous software artifacts (i.e., those containing text and code), such as, programming forum posts (e.g., [9]).

Orthogonal to input and output factors, automatic software summarization approaches are also affected by various purpose factors, including but not limited to:

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5663-3/18/05...$15.00
https://doi.org/10.1145/3183440.3183464

- *Use.* In general, a summary can be used (i) to decide whether to read the whole source document (e.g., [5, 8, 13]); (ii) as a substitute of the original document (e.g., [6, 7, 10, 12]); or (iii) as a preview of the structure/content of the source (e.g., [14]).
- *Summary type.* A summary is *indicative* (e.g., [5, 10, 13]) when it provides a brief description of the main topic of the source without going into details; or *informative* (e.g., [6, 9, 12]) when it provides a more comprehensive (yet short) description of all the ideas in the source.
- *Relation to source.* A summary is *extractive* (e.g., [6, 9, 10, 12, 14]) when it consists only of unprocessed units (e.g., clauses or code statements) selected from the source. If instead the summary includes units that are not an exact part of the source, it is *abstractive* (e.g., [5, 7, 8, 13]).

The main challenges faced by researchers moving forward include selecting and adapting the most appropriate text summarization techniques to work on mixed artifacts, inventing new techniques for code, and generating different summaries of the same artifacts, for different tasks.

2.2 Evaluation of Software Summarization

Assessing the quality of automatically generated summaries represents one of the main challenges in the field. Similar to automated text summarization, more than one acceptable summary might exist for the same source(s), since the summary quality depends on several factors.

Different strategies have been used to evaluate the summarization results. They can be categorized according the human involvement in the process (online vs. offline evaluation) and the quality attribute being assessed (intrinsic vs. extrinsic evaluation). *Online* strategies (e.g., [5-8, 12, 13]) require humans judging the summaries, whereas *offline* strategies (e.g., [9, 10, 14]) require a gold standard to use as the summaries' baseline of comparison. At the same time, *intrinsic* strategies (e.g., [5-7, 9, 10, 12, 14]) evaluate internal properties of a summary, while *extrinsic* strategies (e.g., [6, 8, 12]) evaluate the impact of the summaries when performing a particular task. Researchers have employed all types of evaluations: direct (i.e., online+intrinsic); task-based (i.e., online+extrinsic); target-based (i.e., offline+intrinsic); and automatic (i.e., offline+extrinsic).

A challenge for the research community is developing benchmarks and standards for extrinsic and online evaluations.

3 INTEREST TO THE SE COMMUNITY

There is more and more relevant information available for helping software stakeholders in performing their daily tasks. However, using it may lead to information overload due to its size and complexity. Automated software summarization promises to help with this problem.

We have surveyed the literature on automatic software summarization and found more than 70 publications on this topic, all published in the past ten years. The existing software summarization research shows its potential usefulness in software engineering practice (e.g., [11]) and highlights many challenges, such as, the ones posed by hybrid software artifacts, context-agnostic approaches, and evaluation costs and standards. While the field is established, it is not yet mature, and we did not find records of systematic industry adoption. We expect this to occur as the research in the field advances.

4 AUTHORS' BIOGRAPHIES

Laura Moreno (www.cs.colostate.edu/~lmorenoc) is an Assistant Professor in the Department of Computer Science at Colorado State University. She is particularly interested in software maintenance and evolution research. Her work leverages information contained in various software artifacts through techniques that are at the intersection of software engineering, information retrieval, natural language processing, data mining, and machine learning. An instance of this work is her dissertation, which focused on the generation of software documentation through summarization of source code artifacts.

Andrian Marcus (www.utdallas.edu/~amarcus/) is a Professor in the Department of Computer Science at The University of Texas at Dallas. His current research interests are in software engineering, with focus on software evolution and program comprehension. He is best known for his work on using information retrieval and text mining techniques for software analysis to support comprehension during software evolution.

REFERENCES

[1] Haiduc, S., Aponte, J., Moreno, L., and Marcus, A., "On the Use of Automated Text Summarization Techniques for Summarizing Source Code", in *Proc. of 17th IEEE Working Conf. on Reverese Engineering*, 2010, pp. 35-44.
[2] ANSI, "Guidelines for Abstracts", vol. ANSI/NISO Z39.14-1997 (R2015). Baltimore, Maryland, U.S.A.: NISO Press, 2015.
[3] Jones, K. S., "Automatic summarizing: factors and directions", *Advances in automatic text summarization* 1999, pp. 1-12.
[4] Mani, I. and Maybury, M. T., *Advances in automatic text summarization*, MIT Press, 1999.
[5] Moreno, L., Aponte, J., Sridhara, G., Marcus, A., Pollock, L., and Shanker, V., "Automatic Generation of Natural Language Summaries for Java Classes", in *Proc. of 21st Intl. Conf. on Program Comprehension*, 2013, pp. 23-32.
[6] Moreno, L., Bavota, G., Penta, M. D., Oliveto, R., and Marcus, A., "How can I use this method?", in *Proc. of the 37th Intl. Conf. on Software Eng.* 2015, pp. 880-890.
[7] Moreno, L., Bavota, G., Penta, M. D., Oliveto, R., Marcus, A., and Canfora, G., "ARENA: An Approach for the Automated Generation of Release Notes", *IEEE Transactions on Software Eng.*, vol. 43, no. 2, 2016, pp. 106-127.
[8] Panichella, S., Panichella, A., Beller, M., Zaidman, A., and Gall, H. C., "The impact of test case summaries on bug fixing performance: an empirical investigation", in *Proc. of the 38th Intl. Conf. on Software Eng.*. 2016, pp. 547-558.
[9] Ponzanelli, L., Mocci, A., and Lanza, M., "Summarizing complex development artifacts by mining heterogeneous data", in *Proc. of the 12th Working Conf. on Mining Software Repositories.* Florence, 2015, pp. 401-405.
[10] Rastkar, S., Murphy, G. C., and Murray, G., "Summarizing Software Artifacts: A Case Study of Bug Reports", in *Proc. of 32nd ACM/IEEE Intl. Conf. on Software Eng.*, 2010, pp. 505-514.
[11] Robillard, M. P., Marcus, A., Treude, C., Bavota, G., Chaparro, O., Ernst, N., Gerosa, M. A., Godfrey, M., Lanza, M., Linares-Vásquez, M., Murphy, G. C., Moreno, L., Shepherd, D., and Wong, E., "On-demand Developer Documentation", in *Proc. of 2017 IEEE Intl. Conf. on Software Maintenance and Evolution*, 2017, pp. 479-483.
[12] Sorbo, A. D., Panichella, S., Alexandru, C. V., Shimagaki, J., Visaggio, C. A., Canfora, G., and Gall, H. C., "What would users change in my app? summarizing app reviews for recommending software changes", in *Proc. of the 2016 24th ACM SIGSOFT Intl. Symposium on Foundations of Software Eng.* 2016, pp. 499-510.
[13] Sridhara, G., Hill, E., Muppaneni, D., Pollock, L., and Vijay-Shanker, K., "Towards Automatically Generating Summary Comments for Java Methods", in *Proc. of 25th IEEE/ACM Intl. Conf. on Automated Software Eng.*, 2010, pp. 43-52.
[14] Ying, A. T. T. and Robillard, M. P., "Code fragment summarization", in *Proc. of 9th Joint Meeting on Foundations of Software Engineering.* 2013, pp. 655-658.