Programming with “Big Code”

Martin Vechev
ETH Zurich
Switzerland
martin.vechev@inf.ethz.ch

Eran Yahav
Technion, Haifa
Israel
yahave@cs.technion.ac.il

the essence of knowledge
Boston — Delft

Foundations and Trends® in Programming Languages

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

M. Vechev and E. Yahav. *Programming with "Big Code"*. Foundations and Trends® in Programming Languages, vol. 3, no. 4, pp. 231–284, 2016.

This Foundations and Trends® issue was typeset in LaTeX using a class file designed by Neal Parikh. Printed on acid-free paper.

ISBN: 978-1-68083-231-0

Foundations and Trends® in Programming Languages
Volume 3, Issue 4, 2016

Editorial Board

Editorial Scope

Topics

Foundations and Trends® in Programming Languages publishes survey and tutorial articles in the following topics:

- Abstract interpretation
- Compilation and interpretation techniques
- Domain specific languages
- Formal semantics, including lambda calculi, process calculi, and process algebra
- Language paradigms
- Mechanical proof checking
- Memory management
- Partial evaluation
- Program logic
- Programming language implementation
- Programming language security
- Programming languages for concurrency
- Programming languages for parallelism
- Program synthesis
- Program transformations and optimizations
- Program verification
- Runtime techniques for programming languages
- Software model checking
- Static and dynamic program analysis
- Type theory and type systems

Information for Librarians

Foundations and Trends® in Programming Languages, 2016, Volume 3, 4 issues. ISSN paper version 2325-1107. ISSN online version 2325-1131. Also available as a combined paper and online subscription.

Foundations and Trends® in Programming Languages
Vol. 3, No. 4 (2016) 231–284

DOI: 10.1561/2500000028

Programming with "Big Code"

Martin Vechev
ETH Zurich
Switzerland
martin.vechev@inf.ethz.ch

Eran Yahav
Technion, Haifa
Israel
yahave@cs.technion.ac.il

Contents

Abstract

The vast amount of code available on the web is increasing on a daily basis. Open-source hosting sites such as GitHub contain billions of lines of code. Community question-answering sites provide millions of code snippets with corresponding text and metadata. The amount of code available in executable binaries is even greater. Collectively, these increasing amounts of code have been referred to as "Big Code". In this monograph, we cover some of the recent research trends on leveraging "Big Code" for performing various programming tasks that are difficult to accomplish with traditional techniques.

M. Vechev and E. Yahav. *Programming with "Big Code"*. Foundations and Trends® in Programming Languages, vol. 3, no. 4, pp. 231–284, 2016.

DOI: 10.1561/2500000028.

Full text available at: http://dx.doi.org/10.1561/2500000028

1

Introduction

The vast amount of code available on the web is increasing on a daily basis. Open-source hosting sites such as GitHub contain billions of lines of code. Community question-answering sites provide millions of code snippets with corresponding text and metadata. The amount of code available in executable binaries is even greater. Collectively, these increasing amounts of code have been referred to as "Big Code". In this monograph, we cover some of the recent research trends on leveraging "Big Code" for performing various programming tasks that are difficult to accomplish with traditional techniques.

Size of "Big Code".

As of January 2016, the GitHub open-source platform hosts over 30 million public repositories. The popular question-answering site Stack Overflow contains over 19 million answers, many of them containing code. Both of these sites, as well as other similar sites, are growing rapidly. For example, Fig. 1.1 shows the growth rate in the number of public repositories on GitHub. Looking beyond source code, the amount of binary executables available for download is also increasing rapidly (both in desktop applications and in mobile apps).

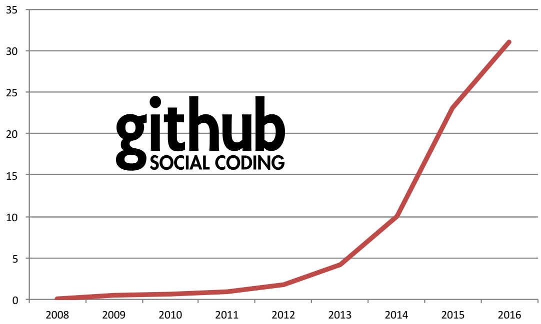

Figure 1.1: Number of public repositories on GitHub (millions).

The availability of these massive amounts of code and meta-data have the potential to revolutionize the way software is being developed, taught, debugged and analyzed. Ideas such as *example-centric programming* Brandt et al. [2010] or *opportunistic programming* Brandt et al. [2009], where the machine assists the programmer in leveraging existing code repositories are not new. They are based on the insight that finding, checking, and adapting an existing solution is easier than creating a solution from scratch. Indeed, various recommendation systems, and code completion tools Thummalapenta and Xie, Zhong et al., Alnusair et al. [2010], Shoham et al. [2007a], Holmes et al., Mandelin et al. [2005], Gvero et al. [2011], Perelman et al. can improve programmer productivity and increase code reuse. While these techniques have made significant progress, they have limited or no support for reusing existing codebases.

Big Code vs. Big Data

Learning from "Big Code" inherits all of the challenges associated with learning from "Big Data" (e.g., images, videos) (see Jagadish et al. [2014]). However, it introduces an additional set of unique problems specific to the domain of programs. The reason is that unlike a traditional data sample (e.g., an image), a data sample in "Big Code" is a program that can represent an *infinite* number of behaviors. Further, it is well known that many seemingly basic questions about programs are in fact undecidable and thus the exact program semantics (that one has to learn from) need to be approximated.

Extracting Semantic Information via Static Analysis

Fortunately, in the past decade, automatic tools for reasoning about programs have advanced to the point where they can be applied to realistic software. These advances have led to impressive success stories in program analysis (e.g., ASTREE,Infer,SLAM). In the context of learning from "Big Code", static analysis is useful for automatically extracting semantic properties of a program (e.g., how a program uses a library, numerical invariants, etc.). This semantic information allows us to reason about a program more precisely than if we had simply treated the program as text or a sequence of tokens. The importance of working with semantic program representations when learning from "Big Code" is supported experimentally by several recent works Raychev et al. [2015, 2014a], Mishne et al. [2012].

Capturing Program Commonalities via Statistical Learning

Recent advances in statistical learning algorithms enable their application to large amounts of semantic information extracted from (a potentially large number of) programs. Algorithms ranging from simple classification up to advanced structured prediction with conditional random fields (CRFs) Sutton and McCallum [2011] can be used to learn complex facts about programs, e.g., predicting semantic properties Raychev et al. [2015]. Similarly, techniques from natural language processing can also be applied to program descriptions. For instance, sites such as `stackoverflow.com` provide a valuable correlation between succinct technical textual descriptions and code snippets. Analyzing a large number of code snippets (using static analysis) and their associated textual descriptions (using NLP) can lead to new approaches for establishing program similarity Zilberstein and Yahav [2016], and automatic program explanation.

Outline

The rest of this manuscript is organized as follows. In Chapter 2, we discuss several recent applications, describe the dimensions of the problem, and illustrate how each application and example system fits these dimensions (we only briefly discuss the technical machinery). Then, in Chapter 3, we select one of the applications (statistical code completion) that captures various aspects of the problem of learning from "Big Code" and illustrate each of these aspects in detail.

References

Tutorial for android. http://www.tutorialforandroid.com/2009/01/changing-screen-brightness.html, a.

Android-er. http://android-er.blogspot.ch/2011/03/set-wallpaper-using-wallpapermanager.html, b.

Android how-to's. https://sites.google.com/site/androidhowto/how-to-1/display-a-web-page, c.

Tutorial for android. http://www.tutorialforandroid.com/2009/10/turn-off-turn-on-wifi-in-android-using.html, d.

Android official documentation, MediaRecorder. http://goo.gl/PZRic.

Stack overflow. http://www.stackoverflow.com/.

Vogella tutorials. http://www.vogella.com/articles/AndroidMedia/article.html.

Miltiadis Allamanis, Earl T. Barr, Christian Bird, and Charles Sutton. Suggesting accurate method and class names. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, pages 38–49, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3675-8. URL http://doi.acm.org/10.1145/2786805.2786849.

Miltiadis Allamanis, Hao Peng, and Charles A. Sutton. A convolutional attention network for extreme summarization of source code. *CoRR*, abs/1602.03001, 2016. URL http://arxiv.org/abs/1602.03001.

A. Alnusair, Tian Zhao, and E. Bodden. Effective API navigation and reuse. In *IRI*, pages 7 –12, aug. 2010.

Glenn Ammons, Rastislav Bodík, and James R. Larus. Mining specifications. In *POPL '02*, 2002. ISBN 1-58113-450-9. URL http://doi.acm.org/10.1145/503272.503275.

Nels E. Beckman, Duri Kim, and Jonathan Aldrich. An empirical study of object protocols in the wild. In *Proceedings of the 25th European Conference on Object-oriented Programming*, ECOOP'11, pages 2–26. Springer-Verlag, 2011. URL http://dl.acm.org/citation.cfm?id=2032497.2032501.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944966.

Benjamin Bichsel, Veselin Raychev, Petar Tsankov, and Martin T. Vechev. Statistical deobfuscation of android applications. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 343–355, 2016. URL http://doi.acm.org/10.1145/2976749.2978422.

Pavol Bielik, Veselin Raychev, and Martin T. Vechev. PHOG: probabilistic model for code. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 2933–2942, 2016. URL http://jmlr.org/proceedings/papers/v48/bielik16.html.

Joel Brandt, Philip J Guo, Joel Lewenstein, Mira Dontcheva, and Scott R Klemmer. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1589–1598. ACM, 2009.

Joel Brandt, Mira Dontcheva, Marcos Weskamp, and Scott R Klemmer. Example-centric programming: integrating web search into the development environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 513–522. ACM, 2010.

Jonathan E. Cook and Alexander L. Wolf. Discovering models of software processes from event-based data. *ACM Trans. Softw. Eng. Methodol.*, 7(3): 215–249, 1998. ISSN 1049-331X.

Barthélémy Dagenais and Laurie J. Hendren. Enabling static analysis for partial Java programs. In *OOPSLA'08*, pages 313–328. ISBN 978-1-60558-215-3. URL http://doi.acm.org/10.1145/1449764.1449790.

Yaniv David, Nimrod Partush, and Eran Yahav. Statistical similarity of binaries. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '16, pages 266–280, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4261-2. URL http://doi.acm.org/10.1145/2908080.2908126.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. . URL http://groups.lis.illinois.edu/amag/langev/paper/elman90findingStructure.html.

Sumit Gulwani. Dimensions in program synthesis. In *symp. on Principles and practice of declarative programming*, PPDP '10, 2010. URL http://doi.acm.org/10.1145/1836089.1836091.

Tihomir Gvero, Viktor Kuncak, and Ruzica Piskac. Interactive synthesis of code snippets. In *Proceedings of the 23rd international conference on Computer aided verification*, CAV'11, pages 418–423, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-22109-5. URL http://dl.acm.org/citation.cfm?id=2032305.2032338.

Tihomir Gvero, Viktor Kuncak, Ivan Kuraj, and Ruzica Piskac. Complete completion using types and weights. In *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '13, Seattle, WA, USA, June 16-19, 2013*, pages 27–38, 2013.

Abram Hindle, Earl T. Barr, Zhendong Su, Mark Gabel, and Premkumar Devanbu. On the naturalness of software. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, pages 837–847, Piscataway, NJ, USA, 2012. IEEE Press. ISBN 978-1-4673-1067-3. URL http://dl.acm.org/citation.cfm?id=2337223.2337322.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Reid Holmes and Gail C. Murphy. Using structural context to recommend source code examples. In *Proceedings of the 27th International Conference on Software Engineering*, ICSE '05, pages 117–125, New York, NY, USA, 2005. ACM. ISBN 1-58113-963-2. URL http://doi.acm.org/10.1145/1062455.1062491.

Reid Holmes, Robert J. Walker, and Gail C. Murphy. Strathcona example recommendation tool. In *FSE'05*, pages 237–240. ISBN 1-59593-014-0. . URL http://doi.acm.org/10.1145/1081706.1081744.

H. V. Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M. Patel, Raghu Ramakrishnan, and Cyrus Shahabi. Big data and its technical challenges. *Commun. ACM*, 57(7):86–94, July 2014. ISSN 0001-0782. URL http://doi.acm.org/10.1145/2611567.

Svetoslav Karaivanov, Veselin Raychev, and Martin T. Vechev. Phrase-based statistical translation of programming languages. In *Onward! 2014, Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software, part of SPLASH '14, Portland, OR, USA, October 20-24, 2014*, pages 173–184, 2014. URL http://doi.acm.org/10.1145/2661136.2661148.

Omer Katz, Ran El-Yaniv, and Eran Yahav. Estimating types in binaries using predictive modeling. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '16, pages 313–326. ACM, 2016. URL http://doi.acm.org/10. 1145/2837614.2837674.

Slava M Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(3):400–401, 1987.

Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume I, May 1995.

Stefan Kombrink, Tomas Mikolov, Martin Karafiát, and Lukás Burget. Recurrent neural network based language modeling in meeting recognition. In *INTERSPEECH*, pages 2877–2880, 2011.

David Mandelin, Lin Xu, Rastislav Bodík, and Doug Kimelman. Jungloid mining: Helping to navigate the api jungle. In *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Imple-mentation*, PLDI '05, pages 48–61. ACM, 2005. URL http://doi.acm. org/10.1145/1065010.1065018.

Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocky. Strategies for training large scale neural network language models. In *ASRU 2011*. IEEE Signal Processing Society, 2011. ISBN 978-1-4673-0366-8.

Alon Mishne, Sharon Shoham, and Eran Yahav. Typestate-based semantic code search over partial programs. In *OOPSLA '12*, 2012. I URL http://doi.acm.org/10.1145/2384616.2384689.

Daniel Perelman, Sumit Gulwani, Thomas Ball, and Dan Grossman. Type-directed completion of partial expressions. In *Proc. of the ACM SIGPLAN conference on Programming Language Design and Implementation*, PLDI '12, pages 275–286. ISBN 978-1-4503-1205-9. URL http://doi.acm. org/10.1145/2254064.2254098.

Veselin Raychev, Martin Vechev, and Eran Yahav. Code completion with statistical language models. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '14, pages 419–428, New York, NY, USA, 2014a. ACM. URL http://doi.acm.org/10.1145/2594291.2594321.

Veselin Raychev, Martin Vechev, and Eran Yahav. Code completion with statistical language models. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, page 44. ACM, 2014b.

Veselin Raychev, Martin Vechev, and Andreas Krause. Predicting program properties from "big code". In *Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '15, pages 111–124. ACM, 2015. ISBN 978-1-4503-3300-9. . URL http://doi.acm.org/10.1145/2676726.2677009.

Veselin Raychev, Pavol Bielik, Martin Vechev, and Andreas Krause. Learning programs from noisy data. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL 2016, pages 761–774, New York, NY, USA, 2016a. ACM. ISBN 978-1-4503-3549-2. . URL http://doi.acm.org/10.1145/2837614.2837671.

Veselin Raychev, Pavol Bielik, and Martin T. Vechev. Probabilistic model for code with decision trees. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Lan-guages, and Applications, OOPSLA 2016*, pages 731–747, 2016b. URL http://doi.acm.org/10.1145/2983990.2984041.

Steven P. Reiss. Semantics-based code search. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 243–253, Washington, DC, USA, 2009. IEEE Computer Society. URL http://dx.doi.org/10.1109/ICSE.2009.5070525.

Ronald Rosenfeld. Two decades of statistical language modeling: Where do we go from here. In *Proceedings of the IEEE*, page 2000, 2000.

Sharon Shoham, Eran Yahav, Stephen Fink, and Marco Pistoia. Static specification m ining u sing a utomata-based a bstractions. I n *Proceedings of the 2007 International Symposium on Software Testing and Analysis*, ISSTA '07, pages 174–184, New York, NY, USA, 2007a. ACM. URL http://doi.acm.org/10.1145/1273463.1273487.

Sharon Shoham, Eran Yahav, Stephen Fink, and Marco Pistoia. Static specification mining using automata-based a bstractions. In *ISSTA '07: Proceedings of the 2007 international symposium on Software testing and analysis*, pages 174–184, 2007b. ISBN 978-1-59593-734-6.

Armando Solar-Lezama. The sketching approach to program synthesis. In *APLAS '09*, 2009. ISBN 978-3-642-10671-2. . URL http://dx.doi.org/10.1007/978-3-642-10672-9_3.

Armando Solar-Lezama, Liviu Tancau, Rastislav Bodik, Sanjit Seshia, and Vijay Saraswat. Combinatorial sketching for finite p rograms. I n *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XII, pages 404–415, New York, NY, USA, 2006. ACM. ISBN 1-59593-451-0. URL http://doi.acm.org/10.1145/1168857.1168907.

Saurabh Srivastava, Sumit Gulwani, and Jeffrey S. Foster. From program verification to program synthesis. In *POPL '10*, 2010. ISBN 978-1-60558-479-9. URL http://doi.acm.org/10.1145/1706299.1706337.

Bjarne Steensgaard. Points-to analysis in almost linear time. In *Proceedings of the 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '96, pages 32–41. ACM, 1996. ISBN 0-89791-769-3. URL http://doi.acm.org/10.1145/237721.237727.

A. Stolcke. SRILM-an Extensible Language Modeling Toolkit. *International Conference on Spoken Language Processing*, 2002.

Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *Machine Learning*, 4(4):267–373, 2011.

Suresh Thummalapenta and Tao Xie. PARSEWeb: a programmer assistant for reusing open source code on the web. In *ASE'07*, pages 204–213. ISBN 978-1-59593-882-4.

R. Vallée-Rai et al. Soot - a Java Optimization Framework. In *Proceedings of CASCON 1999*, pages 125–135, 1999. URL www.sable.mcgill.ca/publications.

Martin Vechev and Eran Yahav. Deriving linearizable fine-grained concurrent objects. In *PLDI '08*, 2008. ISBN 978-1-59593-860-2. URL http://doi.acm.org/10.1145/1375581.1375598

Andrzej Wasylkowski and Andreas Zeller. Mining temporal specifications from object usage. In *Autom. Softw. Eng.*, volume 18, 2011. URL http://dx.doi.org/10.1007/s10515-011-0084-1.

Westley Weimer and George C. Necula. Mining temporal specifications for error detection. In *Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'05, pages 461–476. Springer-Verlag, 2005. URL http://dx.doi.org/10.1007/978-3-540-31980-1_30.

Ian H. Witten and Timothy C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, 1991. URL http://dblp.uni-trier.de/db/journals/tit/tit37.html#WittenB91.

Shir Yadid and Eran Yahav. Extracting code from programming tutorial videos. In *Onward! 2016*, pages 98–111, New York, NY, USA, 2016. ISBN 978-1-4503-4076-2. URL http://doi.acm.org/10.1145/2986012.2986021.

Jinlin Yang, David Evans, Deepali Bhardwaj, Thirumalesh Bhat, and Manuvir Das. Perracotta: mining temporal API rules from imperfect traces. In *ICSE '06*, pages 282–291. ISBN 1-59593-375-1.

Kuat Yessenov, Zhilei Xu, and Armando Solar-Lezama. Data-driven synthesis for object-oriented frameworks. In *OOPSLA '11*, 2011. URL http://doi.acm.org/10.1145/2048066.2048075.

Hao Zhong, Tao Xie, Lu Zhang, Jian Pei, and Hong Mei. MAPO: Mining and recommending API usage patterns. In *ECOOP'09*. ISBN 978-3-642-03012-3. URL http://dx.doi.org/10.1007/978-3-642-03013-0_15.

Meital Zilberstein and Eran Yahav. Leveraging a corpus of natural language descriptions for program similarity. In *Onward! 2016*, pages 197–211, 2016. ISBN 978-1-4503-4076-2. URL http://doi.acm.org/10.1145/2986012.2986013.