Automated Generation of Realistic Test Inputs for Web APIs

Juan C. Alonso
SCORE Lab, Universidad de Sevilla, Seville, Spain

ABSTRACT

Testing web APIs automatically requires generating input data values such as addressess, coordinates or country codes. Generating meaningful values for these types of parameters randomly is rarely feasible, which means a major obstacle for current test case generation approaches. In this paper, we present ARTE, the first semantic-based approach for the Automated generation of Realistic TEst inputs for web APIs. Specifically, ARTE leverages the specification of the API under test to search for meaningful test inputs for the API parameters in knowledge bases like DBpedia. Our approach has been integrated into RESTest, a state-of-the-art tool for API testing, achieving an unprecedented level of automation which allows to generate up to 100% more valid API calls than existing fuzzing techniques, 30% on average. Evaluation results on a set of 26 real-world APIs show that ARTE can generate realistic inputs for 7 out of every 10 parameters, outperforming related approaches.

CCS CONCEPTS

• **Software and its engineering** → **Software testing and debugging**; • **Information systems** → **RESTful web services**.

KEYWORDS

Web of Data, knowledge base, realistic test input, automated testing, RESTful API

ACM Reference Format:
Juan C. Alonso. 2021. Automated Generation of Realistic Test Inputs for Web APIs. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21), August 23–28, 2021, Athens, Greece.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3468264.3473491

1 RESEARCH PROBLEM AND MOTIVATION

Web *Application Programming Interfaces (APIs)* allow heterogeneous software systems to interact over the network [16, 20]. Companies such as Google, Microsoft and Apple provide web APIs to allow other systems to interact with them programmatically. Most web APIs adhere to the REpresentational State Transfer (REST) architectural state, being known as RESTful APIs [14]. RESTful APIs are commonly specified using languages such as the OpenAPI Specification (OAS) [2], which defines a standard notation for describing *RESTful APIs* in terms of operations, input parameters, and expected outputs, among others.

Testing web APIs adequately requires using realistic test inputs such as country names, codes, coordinates, or addresses. As an example, the `find-by-address` operation in the DHL Location Finder API [1] requires users to provide valid ISO country codes (e.g., "`DE`" for Germany), postal codes (e.g., "`10117`"), street addresses (e.g., "`Friedrichstrasse 163`") and localities (e.g., "`Berlin`"). Generating meaningful values for such parameters requires analyzing their name or description. In most cases, generating these values randomly is infeasible, especially when using a black box testing approach, resulting in "client error" responses (4XX status code). To address this issue, most test case generation approaches resort to *data dictionaries*: sets of input values collected by the testers, either manually [19] or, when possible, automatically [22]. This means a major obstacle for automation since data dictionaries must be created and maintained for each non-trivial input parameter, on each API under test.

In this paper, we present an approach for the Automated Generation of Realistic TEst inputs (ARTE) for web APIs. ARTE exploits the specification of the various elements of the API under test by querying knowledges bases to automatically extract realistic test inputs. In contrast to similar approaches, ARTE includes a novel step for the automated inference of regular expressions from previously generated inputs which allows to filter invalid values and to increase the accuracy of the semantic queries. Furthermore, we integrated ARTE into RESTest [19], a state-of-the-art tool for black-box test case generation of RESTful APIs, making our approach fully automated and publicly available.

2 BACKGROUND AND RELATED WORK

The Web of Data [12] is a global data space in continuous growth that contains billions of interlinked queryable data. Resources are identified using URIs and resource relationships are specified using RDF (Resource Description Framework) [15], a standard that defines how to identify relationships between resources in the form of triples composed by a subject, a predicate and an object. The predicate specifies the link that holds between the subject and the object. ARTE uses the SPARQL language [4] to query datasets represented as RDF triples (such as DBpedia [13] or Wikidata [24]).

Our work is the first to leverage the Web of Data for improving test data generation in web APIs. Nevertheless, semantic information retrieval techniques have already been applied in the context of GUI testing. Mariani et al. [18] introduced Link, an approach to generate realistic test inputs for web, desktop and mobile applications. Wanwarang et al. [25] presented SAIGEN, an extension of Link specifically designed for mobile applications. In both cases, test inputs are extracted from a knowledge base (DBpedia) based on the labels associated to the input fields of the GUI. Our work shares similarities with both papers, but also clear differences. On the one hand, web APIs lack GUIs and are intended for developers, rather than for users. This poses new challenges, like the need to resort to other information sources (i.e., the API specification) and

Table 1: Per API breakdown of valid calls generated using fuzzing techniques, SAIGEN and ARTE.

API	Operation	Fuzzing (%)	SAIGEN (%)	ARTE (%)
Amadeus Hotel	Find hotels	13	9.2	**16.7**
Amadeus Hotel	View hotel rooms	44.5	23.4	**60**
Deutschebahn	Get stations	19	27.2	**44.2**
DHL	Find by address	0	0.1	**70**
DHL	Find by coordinates	0	97.9	**100**
OMDb	Search	**40.1**	34.9	35.1
Spotify	Get albums	47.9	49	**95.4**
Spotify	Get album	53.6	48.7	**97.4**
Spotify	Get categories	50	49.7	**70.2**
Spotify	Get category	48.7	51.4	**74.5**
Spotify	Get featured playlists	25.2	25.6	**35.1**
Yelp Fusion	Search businesses	31.4	**50.9**	48.3
Yelp Fusion	Search transactions	52.9	70.6	**86**
Total		32.8	41.4	64.1

to apply different knowledge extraction techniques, such as Natural Language Processing (NLP) techniques [7, 11, 17, 21, 23]. For instance, when the name of a parameter is not descriptive enough (e.g., `'t'` in the OMDb API). On the other hand, we proposed a novel approach to iteratively refine test inputs by automatically generating regular expressions conforming to them.

3 APPROACH AND UNIQUENESS

ARTE is divided into several steps, namely:

Processing test parameters. ARTE receives as inputs the OAS specification and the list of parameters for which it should generate meaningful data (this allows to discard trivial parameters like dates or domain specific identifiers, e.g., a Spotify album ID).

Search for predicates. ARTE applies several matching rules in the parameter name and description to identify keywords to use in the search for predicates in the selected knowledge base, selecting the first predicate that achieves a stablished threshold. For example, ARTE would select the predicate `dbp:webSite` for a parameter named `web_site`.

Search for test inputs. The predicates retrieved during the previous step are used to construct SPARQL queries to search for test inputs. For example, from the predicate `dbp:webSite`, ARTE would obtain a list of URLs corresponding to real websites.

Automated generation of regular expressions. The list of input values may contain entries in different formats (e.g., websites starting with "`https://`" and "`www`"), with only one of them being accepted by the API. ARTE can optionally learn from previous API responses by automatically generating a regular expression that matches only the valid values (e.g., those URLs starting with "`https://`"), this regular expression (generated by leveraging a modified version of RegexGenerator++ [8–10]) is used to filter the list of input values. Optionally, the regular expression can be used to conduct a refined search for input values. ARTE provides the tester with the list of valid and invalid values, enabling both positive and negative testing.

A prototype of ARTE has been implemented in Java leveraging the libraries Jena [5], for the creation of SPARQL queries, Stanford CoreNLP [6], for NLP related tasks, and RegexGenerator++ [8–10], for the generation of regular expressions.

4 RESULTS AND CONTRIBUTIONS

We compared the effectiveness of ARTE with fuzzing techniques and SAIGEN —a related tool for the generation of test inputs for mobile applications [25]. We selected a set of 83 operations belonging to 26 real-world web APIs, 7 of them being industrial APIs with millions of users worldwide. The remaining 19 APIs were selected among the most popular APIs of the RapidAPI repository [3]. We relied on DBpedia as the selected knowledge base, specifically the 2016-10 core dataset[1]. For the preliminary evaluation of our approach, we conducted two experiments, namely:

Experiment 1: Test input generation. For each parameter, we tested the API with 10 randomly selected values among those generated by ARTE. ARTE generated valid inputs for 66.7% of the parameters (66 out of 99), obtaining realistic inputs for 100% of the parameters in 9 out of the 26 APIs, and 50% or more in 19 of them. The automatic generation of regular expressions increased the percentage of realistic test inputs in 4 out of the 13 APIs for which ARTE did not initially achieve a 100% of accepted inputs. On the other hand, SAIGEN only generated realistic values for 38.9% of the parameters, obtaining realistic inputs for 100% of the parameters in 3 APIs, and 50% or more in 11.

Experiment 2: Generation of valid API calls. We compared the effectiveness of ARTE, fuzzing and SAIGEN in generating valid API calls for 13 operations from 6 industrial APIs, depicted in Table 1. For each operation and technique, we generated and executed 1K API calls. The results in Table 1 reveal that ARTE can generate up to 100% more valid API calls than random techniques, 31.3% on average. The highest value of each row is highlighted in boldface, with ARTE achieving the best performance in 11 operations. The application of ARTE with the automated generation of regular expressions yielded an improvement of a 13.5% with respect to its application without this step, obtaining an noticeable improvement in 6 operations. ARTE obtained up to 70% more valid API calls than SAIGEN, with an average improvement of 22.7%, showing that ARTE can significantly improve the results obtained by similar tools. This improvement is mainly due to the automated generation of regular expressions and the NLP techniques applied in both the parameters names and descriptions to obtain keywords for the search for predicates.

The dataset of our preliminary evaluation is available at http://doi.org/10.5281/zenodo.4736860

REFERENCES

[1] 2020. DHL Location Finder API. https://developer.dhl.com/api-reference/location-finder. Accessed December 2020.
[2] 2020. OpenAPI Specification. https://www.openapis.org accessed December 2020.
[3] 2020. RapidAPI API directory. https://rapidapi.com/marketplace. Accessed November 2020.
[4] 2020. SPARQL 1.1 Overview. https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/ Accessed January 2020.
[5] 2021. Apache Jena. https://jena.apache.org/index.html Accessed February 2021.
[6] 2021. Stanford CoreNLP. https://stanfordnlp.github.io/CoreNLP/ Accessed February 2021.
[7] Vimala Balakrishnan and Ethel Lloyd-Yemoh. 2014. Stemming and lemmatization: a comparison of retrieval performances. *Lecture Notes on Software Engineering* (2014), 262–267. https://doi.org/10.7763/LNSE.2014.V2.134

[1] http://downloads.dbpedia.org/2016-10/

[8] Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. 2016. Can a machine replace humans in building regular expressions? A case study. *IEEE Intelligent Systems* 31, 6 (2016), 15–21. https://doi.org/10.1109/MIS.2016.46

[9] Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, Fabiano Tarlao, and Marco Virgolin. 2015. Evolutionary Learning of Syntax Patterns for Genic Interaction Extraction. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*. ACM, 1183–1190. https://doi.org/10.1145/2739480.2754706

[10] A. Bartoli, A. De Lorenzo, E. Medvet, and F. Tarlao. 2016. Inference of Regular Expressions for Text Extraction from Examples. *IEEE Transactions on Knowledge and Data Engineering* 28, 5 (May 2016), 1217–1230. https://doi.org/10.1109/TKDE.2016.2515587

[11] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

[12] C. Bizer, T. Heath, and Tim Berners-Lee. 2009. Linked Data - The Story So Far. *Int. J. Semantic Web Inf. Syst.* 5 (2009), 1–22. https://doi.org/10.4018/jswis.2009081901

[13] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia-A crystallization point for the Web of Data. *Journal of web semantics* 7, 3 (2009), 154–165. https://doi.org/10.1016/j.websem.2009.07.002

[14] Roy Thomas Fielding. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. Ph.D. Dissertation.

[15] Ramanathan Guha and Dan Brickley. 2004. *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation. W3C. https://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[16] Daniel Jacobson, Greg Brail, and Dan Woods. 2011. *APIs: A Strategy Guide*. O'Reilly Media, Inc.

[17] Christopher Manning and Hinrich Schutze. 1999. *Foundations of statistical natural language processing*. MIT press.

[18] Leonardo Mariani, Mauro Pezzè, Oliviero Riganelli, and Mauro Santoro. 2014. Link: Exploiting the Web of Data to Generate Test Inputs. In *Proceedings of the 2014 International Symposium on Software Testing and Analysis*. 373–384. https://doi.org/10.1145/2610384.2610397

[19] Alberto Martin-Lopez, Sergio Segura, and Antonio Ruiz-Cortés. 2020. RESTest: Black-Box Constraint-Based Testing of RESTful Web APIs. In *International Conference on Service-Oriented Computing*. 459–475. https://doi.org/10.1007/978-3-030-65310-1_33

[20] Leonard Richardson, Mike Amundsen, and Sam Ruby. 2013. *RESTful Web APIs*. O'Reilly Media, Inc.

[21] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 conference of the North American chapter of the association for computational linguistics on human language technology-volume 1*. Association for Computational Linguistics, 173–180. https://doi.org/10.3115/1073445.1073478

[22] Emanuele Viglianisi, Michael Dallago, and Mariano Ceccato. 2020. RestTestGen: Automated Black-Box Testing of RESTful APIs. In *International Conference on Software Testing, Verification and Validation*. https://doi.org/10.1109/ICST46399.2020.00024

[23] S Vijayarani, Ms J Ilamathi, and Ms Nithya. 2015. Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks* 5, 1 (2015), 7–16.

[24] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A Free Collaborative Knowledge base. *Commun. ACM* 57, 10 (Sept. 2014), 78–85. https://doi.org/10.1145/2629489

[25] Tanapuch Wanwarang, Nataniel P. Borges, Leon Bettscheider, and Andreas Zeller. 2020. Testing Apps With Real-World Inputs. In *Proceedings of the IEEE/ACM 1st International Conference on Automation of Software Test* (Seoul, Republic of Korea) *(AST '20)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/3387903.3389310