Leveraging Large Language Models to Improve REST API Testing

Myeongsoo Kim
Georgia Institute of Technology
Atlanta, Georgia, USA
mkim754@gatech.edu

Tyler Stennett
Georgia Institute of Technology
Atlanta, Georgia, USA
tstennett3@gatech.edu

Dhruv Shah
Georgia Institute of Technology
Atlanta, Georgia, USA
dshah374@gatech.edu

Saurabh Sinha
IBM Research
Yorktown Heights, New York, USA
sinhas@us.ibm.com

Alessandro Orso
Georgia Institute of Technology
Atlanta, Georgia, USA
orso@cc.gatech.edu

ABSTRACT

The widespread adoption of REST APIs, coupled with their growing complexity and size, has led to the need for automated REST API testing tools. Current tools focus on the structured data in REST API specifications but often neglect valuable insights available in unstructured natural-language descriptions in the specifications, which leads to suboptimal test coverage. Recently, to address this gap, researchers have developed techniques that extract rules from these human-readable descriptions and query knowledge bases to derive meaningful input values. However, these techniques are limited in the types of rules they can extract and prone to produce inaccurate results. This paper presents RESTGPT, an innovative approach that leverages the power and intrinsic context-awareness of Large Language Models (LLMs) to improve REST API testing. RESTGPT takes as input an API specification, extracts machine-interpretable rules, and generates example parameter values from natural-language descriptions in the specification. It then augments the original specification with these rules and values. Our evaluations indicate that RESTGPT outperforms existing techniques in both rule extraction and value generation. Given these promising results, we outline future research directions for advancing REST API testing through LLMs.

CCS CONCEPTS

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

KEYWORDS

Large Language Models for Testing, OpenAPI Specification Analysis

ACM Reference Format:
Myeongsoo Kim, Tyler Stennett, Dhruv Shah, Saurabh Sinha, and Alessandro Orso. 2024. Leveraging Large Language Models to Improve REST API Testing. In *New Ideas and Emerging Results (ICSE-NIER'24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3639476.3639769

1 INTRODUCTION

In today's digital era, web applications and cloud-based systems have become ubiquitous, making REpresentational State Transfer (REST) Application Programming Interfaces (APIs) pivotal elements in software development [29]. REST APIs enable disparate systems to communicate and exchange data seamlessly, facilitating the integration of a wide range of services and functionalities [11]. As their intricacy and prevalence grow, effective testing of REST APIs has emerged as a significant challenge [12, 19, 39].

Automated REST API testing tools (e.g., [3, 5, 6, 9, 14–16, 18, 20, 22, 38]) primarily derive test cases from API specifications [2, 23, 25, 33]. Their struggle to achieve high code coverage [19] often stems from difficulties in comprehending the semantics and constraints present in parameter names and descriptions [1, 17, 19]. To address these issues, assistant tools have been developed. These tools leverage Natural Language Processing (NLP) to extract constraints from parameter descriptions [17] and query parameter names against databases [1], such as DBPedia [7]. However, attaining high accuracy remains a significant challenge for these tools. Moreover, they are limited in the types and complexity of rules they can extract.

This paper introduces RESTGPT, a new approach that harnesses Large Language Models (LLMs) to enhance REST API specifications by identifying constraints and generating relevant parameter values. Given an OpenAPI Specification [25], RESTGPT augments it by deriving constraints and example values. Existing approaches such as NLP2REST [17] require a validation process to improve precision, which involves not just the extraction of constraints but also executing requests against the APIs to dynamically check these constraints. Such a process demands significant engineering effort and a deployed service instance, making it cumbersome and time-consuming. In contrast, RESTGPT achieves higher accuracy without requiring expensive validation. Furthermore, unlike ARTE [1], RESTGPT excels in understanding the context of a parameter name based on an analysis of the parameter description, thus generating more contextually relevant values.

Our preliminary results demonstrate the significant advantage of our approach over existing tools. Compared to NLP2REST without the validation module, our method improves precision from 50% to 97%. Even when compared to NLP2REST equipped with its validation module, our approach still increases precision from 79% to 97%. Additionally, RESTGPT successfully generates both syntactically and semantically valid inputs for 73% of the parameters over the analyzed services and their operations, a considerable

```
/institutions:
  get:
    operationId: searchInstitutions
    produces:
      - application/json
    parameters:
      - name: filters
        in: query
        required: false
        type: string
        description: The filter for the bank search.
            Examples:
            * Filter by State name
            `STNAME:\"West Virginia\"`
            * Filter for any one of multiple State names
            `STNAME:(\"West Virginia\",\"Delaware\")`
      - name: sort_order
        in: query
        required: false
        type: string
        description: Indicator if ascending (ASC) or descending (
            DESC)
    responses:
      '200':
        description: successful operation
        schema:
          type: object
```

Figure 1: A part of FDIC Bank Data's OpenAPI specification.

improvement over ARTE, which could generate valid inputs for 17% of the parameters only. Given these encouraging results, we outline a number of research directions for leveraging LLMs in other ways for further enhancing REST API testing.

2 BACKGROUND AND MOTIVATING EXAMPLE

2.1 REST APIs and OpenAPI Specification

REST APIs are interfaces built on the principles of Representational State Transfer (REST), a design paradigm for networked applications [11]. Designed for the web, REST APIs facilitate data exchange between clients and servers through predefined endpoints primarily using the HTTP protocol [30, 34]. Each client interaction can include headers and a payload, while the corresponding response typically contains headers, content, and an HTTP status code indicating the outcome.

OpenAPI Specification (OAS) [25] is arguably the industry standard for defining RESTful API interfaces. It offers the advantage of machine-readability, supporting automation processes, while also presenting information in a clear, human-readable format. Key features of OAS include the definition of endpoints, the associated HTTP methods, expected input parameters, and potential responses. As an example, Figure 1 shows a portion of the FDIC Bank Data's API specification. This part of the specification illustrates how one might query information about institutions. It also details an expected response, such as the 200 status code, which indicates a successfully processed scenario.

2.2 REST API Testing and Assistant Tools

Automated REST API testing tools [5, 6, 9, 14–16, 18, 20, 22, 38] derive test cases from widely-accepted specifications, primarily OpenAPI [25]. However, these tools often struggle to achieve comprehensive coverage [19]. A significant reason for this is their inability to interpret human-readable parts of the specification [17, 19]. For parameters such as `filters` and `sort_order` shown in Figure 1,

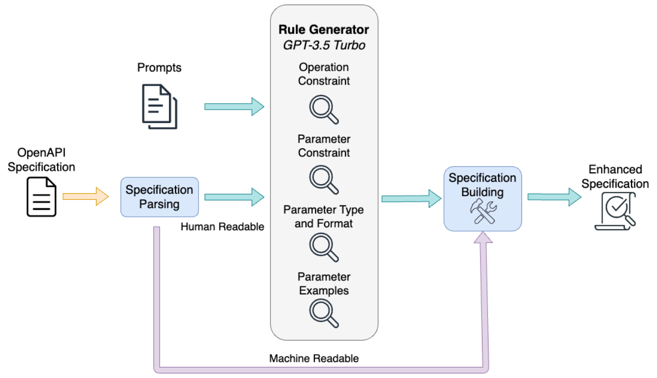

Figure 2: Overview of our approach.

testing tools tend to generate random string values, which are often not valid inputs for such parameters.

In response to these challenges, assistant tools have been introduced to enhance the capabilities of these testing tools. For instance, ARTE [1] taps into DBPedia [7] to generate relevant parameter example values. Similarly, NLP2REST applies natural language processing to extract example values and constraints from descriptive text portions of the specifications [17].

2.3 Large Language Model

Large Language Models (LLMs) [13, 24, 35] represent a transformative leap in the domains of natural language processing (NLP) and Machine Learning. Characterized by their massive size, often containing billions of parameters, these models are trained on vast text corpora to generate, understand, and manipulate human-like text [28]. The architecture behind LLMs are primarily transformer-based designs [36]. Notable models based on this architecture include GPT (Generative Pre-trained Transformer) [27], designed mainly for text generation, and BERT (Bidirectional Encoder Representations from Transformers) [10], which excels in understanding context. These models capture intricate linguistic nuances and semantic contexts, making them adept at a wide range of tasks from text generation to answering questions.

2.4 Motivating Example

The OpenAPI specification for the Federal Deposit Insurance Corporation (FDIC) Bank Data's API, shown in Figure 1, serves to offer insights into banking data. Using this example, we highlight the challenges in parameter value generation faced by current REST API testing assistant tools and illustrate how RESTGPT addresses these challenges.

(1) **Parameter `filters`**: Although the description provides guidance on how the parameter should be used, ARTE's dependency on DBPedia results in no relevant value generation for `filters`. NLP2REST, with its keyword-driven extraction, identifies examples from the description, notably aided by the term "example". Consequently, patterns such as `STNAME: "West Virginia"` and `STNAME: ("West Virginia", "Delaware")` are accurately captured.
(2) **Parameter `sort_order`**: Here, both tools exhibit limitations. ARTE, while querying DBPedia, fetches unrelated values

such as “List of colonial heads of Portuguese Timor”, highlighting its contextual inadequacy. In the absence of identifiable keywords, NLP2REST fails to identify “ASC” or “DESC” as potential values.

In contrast to these tools, RESTGPT is much more effective: with a deeper semantic understanding, RESTGPT accurately discerned that the `filters` parameter was contextualized around state names tied to bank records, and generated test values such as `STNAME: "California"` and multi-state filters such as `STNAME: ("California", "New York")`. Also, it successfully identifies the values “ASC” or “DESC” from the description of the `sort_order` parameter. This example illustrates RESTGPT’s superior contextual understanding, which enable it to outperform the constrained or context-blind methodologies of existing tools.

3 OUR APPROACH

3.1 Overview

Figure 2 illustrates the RESTGPT workflow, which starts by parsing the input OpenAPI specification. During this phase, both machine-readable and human-readable sections of each parameter are identified. The human-readable sections provide insight into four constraint types: operational constraints, parameter constraints, parameter type and format, and parameter examples [17].

The Rule Generator, using a set of crafted prompts, extracts these four rules. We selected GPT-3.5 Turbo as the LLM for this work, given its accuracy and efficiency, as highlighted in a recent report by OpenAI [24]. The inclusion of few-shot learning further refines the model’s output. By providing the LLM with concise, contextually-rich instructions and examples, the few-shot prompts ensure the generated outputs are both relevant and precise [8, 21]. Finally, RESTGPT combines the generated rules with the original specification to produce an enhanced specification.

3.2 Rule Generator

To best instruct the model on rule interpretation and output formatting, our prompts are designed around four core components: guidelines, cases, grammar highlights, and output configurations.

Guidelines

1. Identify the parameter using its name and description.
2. Extract logical constraints from the parameter description, adhering strictly to the provided format.
3. Interpret the description in the least constraining way.

The provided guidelines serve as the foundational instructions for the model, framing its perspective and clarifying its primary objectives. Using the guidelines as a basis, RESTGPT can then proceed with more specific prompting.

Cases

Case 1: If the description is non-definitive about parameter requirements: Output "None".

...

Case 10: For complex relationships between parameters: Combine rules from the grammar.

The implementation of cases in model prompting plays a pivotal role in directing the model’s behaviour, ensuring that it adheres to precise criteria as depicted in the example. Drawing inspiration from Chain-of-Thought prompting [37], we decompose rule extraction into specific, manageable pieces to mitigate ambiguity and, consequently, improve the model’s processing abilities.

Grammar Highlights

Relational Operators: ’<’, ’>’, ’<=’, ’>=’, ’==’, ’! =’
Arithmetic Operators: ’+’, ’−’, ’∗’, ’/’
Dependency Operators: ’AllOrNone’, ’ZeroOrOne’, ...

The Grammar Highlights emphasize key operators and vocabulary that the model should recognize and employ during rule extraction. By providing the model with a fundamental context-specific language, RESTGPT identifies rules within text.

Output Configurations

Example Parameter Constraint: min [minimum], max [maximum], default [default]
Example Parameter Format: type [type], items [item type], format [format], collectionFormat [collectionFormat]

After guiding the model through the rule-extraction process via specific prompting, we lastly define output formatting to compile the model’s findings into a simple structure for subsequent processing.

Additionally, the Rule Generator also oversees the value-generation process, which is executed during the extraction of parameter example rules. Our artifact [31] provides details of all the prompts and their corresponding results.

3.3 Specification Enhancement

The primary objective of RESTGPT is to improve the effectiveness of REST API testing tools. We accomplish this by producing enhanced OpenAPI specifications, augmented with rules derived from the human-readable natural-language descriptions in conjunction with the machine-readable OpenAPI keywords [32].

As illustrated in Figure 2, the *Specification Parsing* stage extracts the machine-readable and human-readable components from the API specification. After rules from the natural language inputs have been identified by the *Rule Generator*, the *Specification Building* phase begins. During this phase, the outputs from the model are processed and combined with the machine-readable components, ensuring that there is no conflict between restrictions. For example, the resulting specification must have the `style` attribute only if the data type is `array` or `object`. The final result is an enriched API specification that contains constraints, examples, and rules extracted from the human-readable descriptions.

4 PRELIMINARY RESULTS

4.1 Evaluation Methodology

We collected nine RESTful services from the NLP2REST study. The motivation behind this selection is the availability of a ground truth

Table 1: Effectiveness of NLP2REST and RESTGPT.

	No. of Rules in	NLP2REST Without Validation Process						NLP2REST With Validation Process						RESTGPT					
REST Service	Ground Truth	TP	FP	FN	Precision	Recall	F_1	TP	FP	FN	Precision	Recall	F_1	TP	FP	FN	Precision	Recall	F_1
FDIC	45	42	36	3	54%	93%	68%	42	25	3	63%	93%	75%	44	0	1	100%	98%	99%
Genome Nexus	81	79	3	2	96%	98%	97%	79	3	2	96%	98%	97%	75	0	6	100%	93%	96%
LanguageTool	20	20	12	0	63%	100%	77%	18	2	2	90%	90%	90%	18	0	3	100%	86%	92%
OCVN	17	15	2	2	88%	88%	88%	13	1	4	93%	76%	84%	15	2	1	88%	94%	91%
OhSome	14	13	66	1	16%	93%	28%	12	11	2	52%	80%	63%	12	3	2	80%	86%	83%
OMDb	2	2	0	0	100%	100%	100%	2	0	0	100%	100%	100%	2	0	0	100%	100%	100%
REST Countries	32	28	1	4	97%	88%	92%	28	0	4	100%	88%	93%	30	0	2	100%	94%	97%
Spotify	88	83	68	5	55%	94%	69%	82	28	6	75%	93%	83%	86	2	4	98%	96%	97%
YouTube	34	30	126	4	19%	88%	32%	28	9	6	76%	82%	79%	24	2	8	92%	75%	83%
Total	**333**	**312**	**314**	**21**	**50%**	**94%**	**65%**	**304**	**79**	**29**	**79%**	**91%**	**85%**	**306**	**9**	**27**	**97%**	**92%**	**94%**

Table 2: Accuracy of ARTE and RESTGPT.

Service Name	ARTE	RESTGPT
FDIC	25.35%	77.46%
Genome Nexus	9.21%	38.16%
Language-Tool	0%	82.98%
OCVN	33.73%	39.76%
OhSome	4.88%	87.80%
OMDb	36.00%	96.00%
REST-Countries	29.66%	92.41%
Spotify	14.79%	76.06%
Youtube	0%	65.33%
Average	16.93%	72.68%

of extracted rules in the NLP2REST work [17]. Having this data, we could easily compare our work with NLP2REST.

To establish a comprehensive benchmark, we incorporated a comparison with ARTE as well. Our approach was guided by the ARTE paper, from which we extracted the necessary metrics for comparison. Adhering to ARTE's categorization of input values as Syntactically Valid and Semantically Valid [1], two of the authors meticulously verified the input values generated by RESTGPT and ARTE. Notably, we emulated ARTE's approach in scenarios where more than ten values were generated by randomly selecting ten from the pool for analysis.

4.2 Results and Discussion

Table 1 presents a comparison of the rule-extraction capabilities of NLP2REST and RESTGPT. RESTGPT excels in precision, recall, and the F_1 score across a majority of the REST services. NLP2REST, while effective, hinges on a validation process that involves evaluating server responses to filter out unsuccessful rules. This methodology demands engineering effort, and its efficacy is constrained by the validator's performance.

In contrast, RESTGPT eliminates the need for such validation entirely with its high precision. Impressively, RESTGPT's precision of 97% surpasses even the precision of NLP2REST post-validation, which stands at 79%. This emphasizes that RESTGPT is able to deliver superior results without a validation stage. This result shows an LLM's superior ability in nuanced rule detection, unlike conventional NLP techniques that rely heavily on specific keywords.

Furthermore, Table 2 presents data on accuracy of ARTE and RESTGPT. The data paint a clear picture: RESTGPT consistently achieves higher accuracy than ARTE across all services. This can be attributed to the context-awareness capabilities of LLMs, as discussed in Section 2. For example, in language-tool service, we found that, for the `language` parameter, ARTE generates values such as "Arabic", "Chinese", "English", and "Spanish". However, RESTGPT understands the context of the language parameter, and generates language code such as "en-US" and "de-DE".

5 FUTURE PLANS

Given our encouraging results on LLM-based rule extraction, we next outline several research directions that we plan to pursue in leveraging LLMs for improving REST API testing more broadly.

Model Improvement. There are two ways in which we plan to create improved models for supporting REST API testing. First, we will perform task-specific fine-tuning of LLMs using data from APIs-guru [4] and RapidAPI [26], which contain thousands of real-world API specifications. We will fine-tune RESTGPT with these datasets, which should enhance the model's capability to comprehend diverse API contexts and nuances. We believe that this dataset-driven refinement will help RESTGPT understand a broader spectrum of specifications and generate even more precise testing suggestions. Second, we will focus on creating lightweight models for supporting REST API testing, such that the models do not require expensive computational resources and can be deployed on commodity CPUs. To this end, we will explore approaches for trimming the model, focusing on retaining the essential neurons and layers crucial for our task.

Improving fault detection. RESTGPT is currently restricted to detecting faults that manifest as 500 server response codes. By leveraging LLMs, we intend to expand the types of bugs that can be detected, such as bugs related to CRUD semantic errors or discrepancies in producer-consumer relationships. By enhancing RESTGPT's fault-finding ability in this way, we aim to make automated REST API testing more effective and useful in practice.

LLM-based Testing Approach. We aim to develop a REST API testing tool that leverages server messages. Although server messages often contain valuable information, current testing tools fail to leverage this information [17]. For instance, if a server hint suggests crafting a specific valid request, RESTGPT, with its semantic understanding, could autonomously generate relevant tests. This would not only enhance the testing process but also ensure that potential loopholes that the server messages may indicate would not be overlooked.

ACKNOWLEDGMENTS

This work was partially supported by NSF, under grant CCF-0725202, DOE, under contract DE-FOA-0002460, and gifts from Facebook, Google, IBM Research, and Microsoft Research.

REFERENCES

[1] J. C. Alonso, A. Martin-Lopez, S. Segura, J. Garcia, and A. Ruiz-Cortes. 2023. ARTE: Automated Generation of Realistic Test Inputs for Web APIs. *IEEE Transactions on Software Engineering* 49, 01 (jan 2023), 348–363. https://doi.org/10.1109/TSE.2022.3150618

[2] API Blueprint. 2023. API Blueprint. https://apiblueprint.org/

[3] Apiary. 2023. Dredd. https://github.com/apiaryio/dredd

[4] APIs.guru. 2023. APIs-guru. https://apis.guru/

[5] Andrea Arcuri. 2019. RESTful API Automated Test Case Generation with EvoMaster. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 28, 1, Article 3 (jan 2019), 37 pages. https://doi.org/10.1145/3293455

[6] Vaggelis Atlidakis, Patrice Godefroid, and Marina Polishchuk. 2019. RESTler: Stateful REST API Fuzzing. In *Proceedings of the 41st International Conference on Software Engineering* (Montreal, Quebec, Canada) *(ICSE '19)*. IEEE Press, Piscataway, NJ, USA, 748–758. https://doi.org/10.1109/ICSE.2019.00083

[7] 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.

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[9] Davide Corradini, Amedeo Zampieri, Michele Pasqua, Emanuele Viglianisi, Michael Dallago, and Mariano Ceccato. 2022. Automated black-box testing of nominal and error scenarios in RESTful APIs. *Software Testing, Verification and Reliability* 32 (01 2022). https://doi.org/10.1002/stvr.1808

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[11] Roy Thomas Fielding. 2000. *Architectural Styles and the Design of Network-Based Software Architectures.* Ph. D. Dissertation. University of California, Irvine.

[12] Amid Golmohammadi, Man Zhang, and Andrea Arcuri. 2023. Testing RESTful APIs: A Survey. *ACM Trans. Softw. Eng. Methodol.* 33, 1, Article 27 (nov 2023), 41 pages. https://doi.org/10.1145/3617175

[13] Google. 2023. Google Bard. https://bard.google.com/

[14] Zac Hatfield-Dodds and Dmitry Dygalo. 2022. Deriving Semantics-Aware Fuzzers from Web API Schemas. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings* (Pittsburgh, Pennsylvania) *(ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 345–346. https://doi.org/10.1145/3510454.3528637

[15] Stefan Karlsson, Adnan Causevic, and Daniel Sundmark. 2020. QuickREST: Property-based Test Generation of OpenAPI-Described RESTful APIs. In *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*. IEEE Press, Piscataway, NJ, USA, 131–141. https://doi.org/10.1109/ICST46399.2020.00023

[16] Stefan Karlsson, Adnan Čaušević, and Daniel Sundmark. 2021. Automatic Property-based Testing of GraphQL APIs. In *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*. IEEE Press, Piscataway, NJ, USA, 1–10. https://doi.org/10.1109/AST52587.2021.00009

[17] Myeongsoo Kim, Davide Corradini, Saurabh Sinha, Alessandro Orso, Michele Pasqua, Rachel Tzoref-Brill, and Mariano Ceccato. 2023. Enhancing REST API Testing with NLP Techniques. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2023)*. Association for Computing Machinery, New York, NY, USA, 1232–1243. https://doi.org/10.1145/3597926.3598131

[18] Myeongsoo Kim, Saurabh Sinha, and Alessandro Orso. 2023. Adaptive REST API Testing with Reinforcement Learning. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE Press, Piscataway, NJ, USA, 446–458. https://doi.org/10.1109/ASE56229.2023.00218

[19] Myeongsoo Kim, Qi Xin, Saurabh Sinha, and Alessandro Orso. 2022. Automated Test Generation for REST APIs: No Time to Rest Yet. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis* (Virtual, South Korea) *(ISSTA 2022)*. Association for Computing Machinery, New York, NY, USA, 289–301. https://doi.org/10.1145/3533767.3534401

[20] Kerry Kimbrough. 2023. Tcases. https://github.com/Cornutum/tcases

[21] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Comput. Surveys* 55, 9 (2023), 1–35.

[22] Alberto Martin-Lopez, Sergio Segura, and Antonio Ruiz-Cortés. 2021. RESTest: Automated Black-Box Testing of RESTful Web APIs. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis* (Virtual, Denmark) *(ISSTA 2021)*. Association for Computing Machinery, New York, NY, USA, 682–685. https://doi.org/10.1145/3460319.3469082

[23] MuleSoft, LLC, a Salesforce company. 2020. RAML. https://raml.org/

[24] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[25] OpenAPI. 2023. OpenAPI standard. https://www.openapis.org.

[26] R Software Inc. 2023. RapidAPI. https://rapidapi.com/terms/

[27] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

[28] Alec Radford, Jeffrey Wu, Dario Amodei, Daniela Amodei, Jack Clark, Miles Brundage, and Ilya Sutskever. 2019. Better language models and their implications.

[29] Leonard Richardson, Mike Amundsen, and Sam Ruby. 2013. *RESTful Web APIs: Services for a Changing World.* O'Reilly Media, Inc., Sebastopol, CA, USA.

[30] Alex Rodriguez. 2008. Restful web services: The basics. *IBM developerWorks* 33, 2008 (2008), 18.

[31] SE@GT. 2023. Experiment infrastructure, data, and results for RESTGPT. https://github.com/selab-gatech/RESTGPT.

[32] SmartBear Software. 2023. OpenAPI data model. https://swagger.io/docs/specification/data-models/keywords/.

[33] SmartBear Software. 2023. Swagger. https://swagger.io/specification/v2/.

[34] Stefan Tilkov. 2007. A brief introduction to REST.

[35] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971 [cs.CL]

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc., Red Hook, NY, USA. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[37] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[38] Huayao Wu, Lixin Xu, Xintao Niu, and Changhai Nie. 2022. Combinatorial Testing of RESTful APIs. In *Proceedings of the 44th International Conference on Software Engineering* (Pittsburgh, Pennsylvania) *(ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 426–437. https://doi.org/10.1145/3510003.3510151

[39] Man Zhang and Andrea Arcuri. 2023. Open Problems in Fuzzing RESTful APIs: A Comparison of Tools. *ACM Trans. Softw. Eng. Methodol.* 32, 6, Article 144 (sep 2023), 45 pages. https://doi.org/10.1145/3597205