Smart Contracts Categorization With Topic Modeling Techniques

Giacomo Ibba[1], Marco Ortu[2], and Roberto Tonelli[1]

[1] University of Cagliari, Department of Mathematics and Computer Science, Cagliari, Italy
[2] University of Cagliari, Department of Business School, Cagliari, Italy
{giacomo.ibba,marco.ortu,roberto.tonelli}@unica.it

Abstract. One of the main advantages of the Ethereum blockchain is the possibility of developing smart contracts in a Turing complete environment. These general-purpose programs provide a higher level of security than traditional contracts and reduce other transaction costs associated with the bargaining practice. Developers use smart contracts to build their tokens and set up gambling games, crowdsales, ICO, and many others. Since the number of smart contracts inside the Ethereum blockchain is several million, it is unthinkable to check every program manually to understand its functionality. At the same time, it would be of primary importance to group sets of Smart Contracts according to their purposes and functionalities. One way to group Ethereum's smart contracts is to use topic modeling techniques, taking advantage of the fact that many programs representing a specific topic are similar in the program structure. Starting from a dataset of 130k smart contracts, we built a Latent Dirichlet Allocation (LDA) model to spot the number of topics within our sample. Computing the coherence values for a different number of topics, we found out that the optimal number was 15. As we expected, most programs are tokens, games, crowdfunding platforms, and ICO.

Keywords: Blockchain · Smart Contract · Ethereum · LDA · Topic Modeling · Ponzi Scheme · Token · ICO · Smart Contracts Trends.

1 Introduction

Over the past few years, the number of smart contracts deployed and verified in the Ethereum blockchain increased exponentially [13]. They provide a higher level of security than traditional contracts offer and reduce costs associated with bargaining practices. Thanks to these particular features, smart contracts are often used for tasks that include management and transfers of currency. Despite being used in particular for developing tokens [1], launching ICOs [8] and crowdfunding platforms [4], developers take advantage of Ethereum's smart contracts to build gambling games, wallets, voting applications, and many others. Sometimes it is also possible to spot scam contracts inside the Ethereum chain, particularly the Ponzi Scheme, a pyramidal scheme in which participants recruit investors with the promise of easy earnings and high-interest rates relative to the initial investment. The first Ethereum contract appearance dates back to

2015, but developers started to exploit this technology widely in 2017. Since then, the number of contracts deployed in the blockchain has reached several million, so it is impossible to manually check each contract deployed since 2017. Given the large number of programs to scan, it is difficult to understand how many different contracts are stored, since many developers adopt the practice to deploy many times the same or very similar contracts, even for testing purposes. One way to analyze such a huge set of smart contracts is to use unsupervised machine learning techniques. It is essential to report and classify these programs correctly for many reasons. For example, inside the Ethereum blockchain it is possible to spot scam contracts. These type of contracts aims to steal Ether from networks participants, thus it is of fundamental importance to identify and report scam contracts properly. Another motivation is for statistical purposes since a massive analysis of smart contracts functionalities is still missing in literature. Furthermore, keeping track of trends, new categories of application, and statistics on the smart contracts' topology leads to a better understanding of the technology in general and in particular of the potential applications. It is already known that developers widely exploit contracts to build their tokens and set up secure money transfer programs. Still it is unknown the number of contracts' categories, and how many samples per category have been deployed from 2016-2017 (when smart contracts started to spread) to nowadays. It would also be interesting to build a taxonomy showing the main types of smart contracts stored inside the Ethereum blockchain. The taxonomy could show which main contract types were present at the beginning of Ethereum's life cycle and the primary categories now, and the smart contracts trend over the years.
Previous works [2] propose a smart contracts taxonomy of five categories.

- **Financial**: contracts managing, gathering, or distributing money.
- **Notary**: contracts that certify ownership and provenance of data.
- **Game**: contracts implementing gambling games and skill games.
- **Wallet**: contracts that, in general, simplify the interaction with the blockchain.
- **Library**: contracts implementing general-purpose operations.

According to the study, the most used smart contracts in Ethereum are financial, libraries and games. The study also gives an idea about the the correlation between design patterns [22] [14] and smart contracts categories, but is essentially quite limited.

2 STATE OF ART

The overgrowing number of Smart Contracts leads to several problems. First, retrieving only a specific smart contracts' category could be complicated. Second, several contracts could also be scams; spotting them and avoiding their use are primary tasks. Often developers tend to deploy several versions of the same program in the blockchain with minor differences. Apart from this, other developers (for experimental and test purposes) deploy updated versions or copies of existing smart contracts, leading to a disproportionate number of similar programs. These problems combined have attracted the attention of many researchers. Indeed several works propose smart contracts classification techniques. One of the several approaches [10] consists in spotting behavior patterns by performing a transaction-based analysis. Another exciting approach [19] combines bi-directional LSTM to capture grammar rules and context information in source

code and Gaussian LDA to generate comment features. Another approach [17] exploits the bytecode instead of source code to classify contracts. Other works are more specific in classification; indeed, researchers carried out several projects to classify smart contracts bugs [7], vulnerabilities [9] [12], and others aiming to spot scam contracts [5]. Considering all previous works, we can say that smart contracts classification is a high-interest task. The problem with previous projects that perform a general classification over contracts is that only a subset of the sample is analyzed; indeed, it is widespread to find a dataset composed of a number between 10.000 and 20.000 contracts. Considering the overall number of contracts within the Ethereum blockchain, we assume that 10.000 programs are insufficient to give us an idea of the actual distribution over smart contracts' categories. In particular, the statement is true for those datasets that include only smart contracts with specific functionality (i.e., money transfer programs) and datasets including random selections of contracts chosen within a definite and short period (i.e., only 2017 or only 2019-2020 programs). Furthermore, some of the proposed models are specific for a precise type of classification, like spotting Ponzi schemes [6]. Our purpose is to find an approach that guarantees us to group smart contracts into categories, starting from a high number of samples, taking advantage of already existing datasets.

3 DATASET

Our dataset is a collection of Solidity smart contracts retrieved from three already existing datasets. The first dataset is SmartBugs [23], which contains vulnerable smart contracts. The second dataset from which we collected our contracts is SmartCorpus [16], an organized, reasoned, and up-to-date repository where developers and researchers can efficiently and systematically access Solidity source code and other metadata about Ethereum smart contracts. The last dataset we exploited is SmartSanctuary [15], an organized repository containing only Ethereum verified smart contracts. What is interesting about this dataset is that it includes smart contract sources for various networks, and it is constantly updated. We will refer to our dataset as SSCB (Smart Sanctuary Corpus Bugs). Our dataset includes contracts within a time frame ranging from 2017 to 2021, of which the most represented years are 2021, 2018, and 2020. Having a lot of such recent contracts increased the representativeness of our dataset since 2018, saw an exponential growth of smart contracts' usage. Nevertheless, we lack a lot of 2019 and all the 2015 and 2016 contracts, so we could enrich our sample and its representativity by adding the programs missed in our dataset. Figure 1 shows the number of samples for each dataset where we also report the data cleaned from duplicates. Some general considerations are needed about the dataset; indeed, before proceeding with our topic modeling, we manually checked about 600 smart contracts to have an idea on which and how many types of smart contracts we were dealing with. Most of the contracts are token contracts, including Non-Fungible-Tokens (NFT) and token exchange programs. Other relevant categories found by manual inspection are Timelock contracts, Crowdsale, ICOs, and gambling games. We were not able to manually categorize some contracts, such as 'Hello World' contracts and others performing nonsense operations. Another exciting sample includes investment contracts, some of which are Ponzi schemes that are immediately recognizable [11]

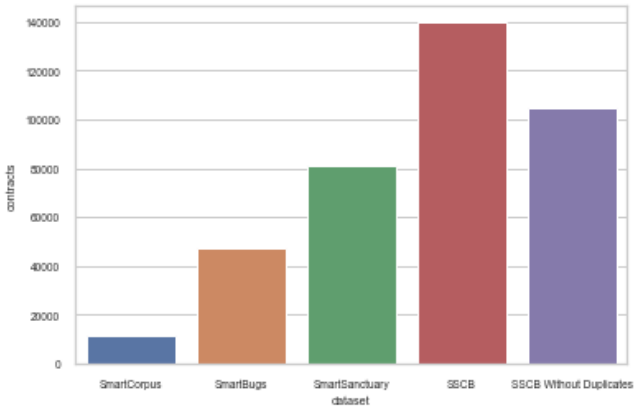

Fig. 1: Number of contracts for each dataset

4 RESEARCH METHODOLOGY

First of all, we combined the three datasets described in the previous session into one, building a CSV with only two features: the contract's address and the source code. We dropped any possible duplicate from the dataset, and we joined all different source codes having the same address since being part of the same contract.
Our work aims to categorize our collection of Ethereum smart contracts which include Solidity source code and comments made by developers. The main idea is that the terms used to set variable names, function names, struct names, and comments could help to spot the contract's purpose. Another interesting observation is that some programs have a well-defined structure in terms of source code; indeed, many developers recycle the source code of already deployed contracts to build their programs. For example, observing tokens, gambling games, investments, and others are pretty much the same looking at the code and structure, and this feature could help in our analysis. Taking advantage of these considerations, we decided to use Natural Language Processing (NLP) techniques to capture all the relevant information from the source code to perform contracts modeling and categorization. In particular, we used topic modeling [20] techniques and the Latent Dirichlet Allocation (LDA) model [3] to categorize our samples. In topic modeling, a task of fundamental importance is the stopwords removal. In the case of our dataset we are not dealing with human language. Still, we are dealing with code language, so one aspect to consider is that all language keywords, apart from a few exceptions, will have the same probability over the categories. Since we have collected only Solidity contracts, we added to the stopword list all the Solidity language keywords. We also added assembly terms since it is possible to write assembly code inside Ethereum smart contracts. Also, we must consider that developers may write smart contracts in snake-case or camel-case. So, we need to pre-process text in such a way as to split words separated by underscores and attached words, in which the first word begins with a lowercase letter, while the following ones with a capital letter. Eventually, we re-

moved new lines characters and characters with exponent. To find the optimal number of topics, we computed the topic coherence, which measures the score of a single topic by measuring the degree of semantic similarity between high-scoring words in the topic. In particular, we computed the c_v measure [18], which is based on a sliding window, one-set segmentation of the top words, and an indirect confirmation measure that uses normalized pointwise mutual information (NPMI) and the cosine similarity. These measurements help distinguish between semantically interpretable topics and topics that are artifacts of statistical inference. After building our LDA model, we found the optimal model by computing the coherence values augmenting the number of topics. Figure 2

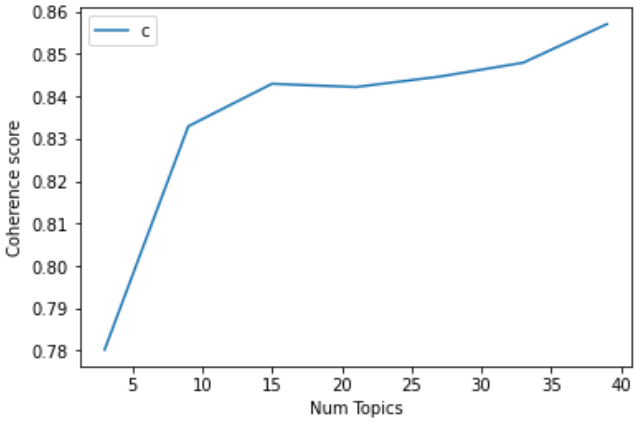

Fig. 2: Coherence score for different number of topics

shows the coherence score for different number of topics. As the graphic shows, the curve peaks at 15 topics, stabilizes, and after 20 topics, it increases again. If the coherence score seems to keep increasing, it could be a reasonable choice to pick the model that gave the highest CV before the curve flattens out or before a major drop.

5 RESULTS

After carrying out our analysis with the LDA model, we were able to find 15 different topics inside our dataset of contracts. The taxonomy in Figure 3 shows the macro-categories and relative sub-categorizations. We detected the following macro-categories, which almost corresponds to those already detected in literature:

- **Financial**
- **Notary**
- **Game**
- **Wallet**
- **Token**

We further expanded this categorization into subcategories. The arrows show which are the relationships between the macro-categories and the sub-categories. For example, Bank, Investment, Crowdsale, ICO, and Ponzi scheme are exclusive topics of the Financial category. Other sub-categories such as Authorization, Termination, and Time Constraint are not topics belonging to a single class; indeed, we can observe these design patterns in several contracts categories. A good example is the Time Constraint topic, a sub-category of the Financial, Wallet, and Game macro-categories. We assigned a document to a particular category by computing each topic's percentage contribution and choosing the topic with the highest percentage. We considered the one proposed in [2] to

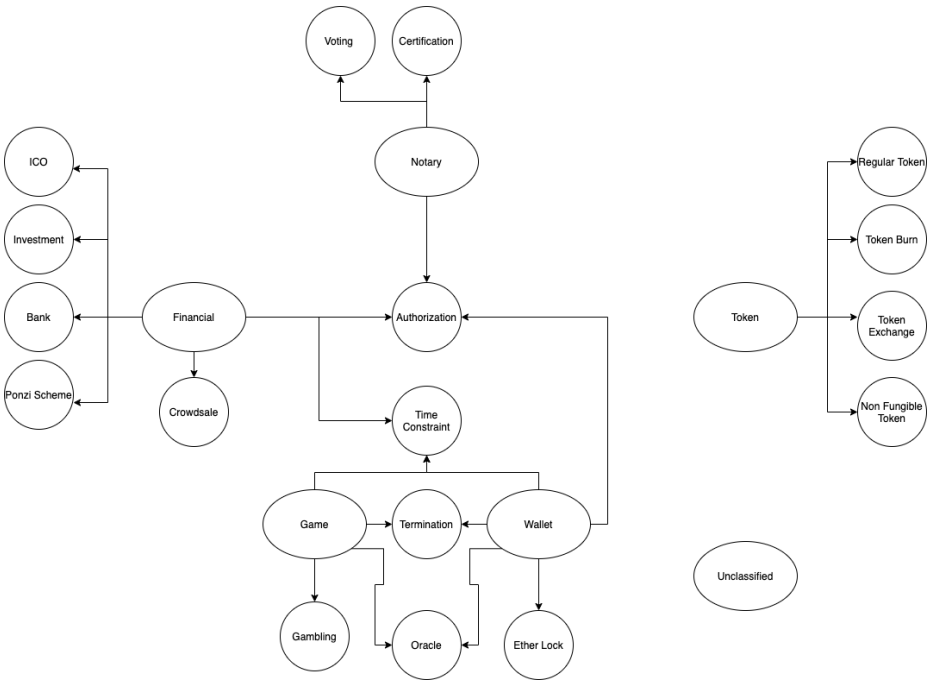

Fig. 3: Smart Contracts Taxonomy

build our taxonomy, but with some crucial differences and further considerations. Also, to build the hierarchy of macro and sub-categories, we considered the already delivered studies about smart contracts design patterns. Furthermore, we added sub-categories that previous studies did not include but detected in our manual check process of contracts (i.e., token burning programs, non-fungible tokens, and Ponzi schemes). The first difference is that we didn't consider Library contracts as a category. It is pretty uncommon nowadays to find a smart contract whose only purpose is to contain a library; indeed, libraries such as 'SafeMath' or similar are most of the time part of more sophisticated programs. We considered the Token design pattern as a separate category.

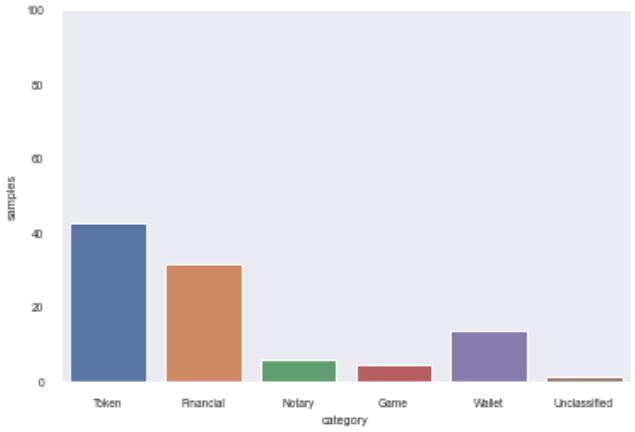

Fig. 4: Distribution of smart contracts categories over SSCB

Indeed, developers can follow several patterns to build their token contracts. The most common pattern is the 'simple token': these programs contain only a token structure and few tokens transfer and burn functions. Another interesting pattern is the Non-Fungible-Token (NFT) [21] one, which certifies a digital asset as unique and therefore not interchangeable. Other common patterns for the token category are programs for exchanging and tokens burning. In this work we assume that all the Token sub-categories are considered in the same topic. For what concerns the Financial class, we found several topics of interest: indeed, some programs aim to store and preserve Ether as a bank. Others are investments oriented programs, and there are also ICO and Crowdfunding platforms. There are two types for what concerns investment programs: the first ones are regular investment programs, while the others are Ponzi schemes, which are pyramidal models, and so only the very first few users to invest their money in these contracts will have profit. Other unique topics are 'gambling' and 'ether lock,' which are Game and Wallet categories topics. The first pattern concerns gambling games, such as roulettes, card games, and others. The topic 'ether lock' instead involves locking and storing Ether for a certain amount of time; they are different from bank programs because these last allow users to retrieve their money at any time. Other topics are common to several categories, like time constraints, used by Financial, Game, and Wallet contracts. Another unique type is the Notary one, which includes contracts for voting applications and certification programs. Also, we spotted several contracts that we couldn't categorize as 'Hello World' programs and other particular types of contracts.These programs were assigned with the topic with the lowest contribution percentage and are identified by incoherent keywords, especially if compared with the other categories.
Regarding the distribution of smart contracts in SSCB, as we expected, most of the programs belong to Tokens and Financial categories, covering most of 70% of our dataset. In contrast, the others cover the remaining part of SSCB. Figure 4 summarizes cate-

Category	Topics	Top 5 Keywords
Token	Regular Token NFT Token Burn Token Exchange	buyuser selluser buyuseraddruser buystorageuser buydatauser
Game	Gambling Oracle Termination Time Constraint	kill uturn validator empty game
Notary	Authorization Certification Voting	vote receivevote destroy paramvalparam pressguy
Wallet	Ether Lock Oracle Termination Authorization Time Constraint	destroy block kill time takeether
Financial	Bank Investment Ponzi Scheme Time Constraint Authorization ICO Crowdsale	eaterfunction withdraw payonlyagency empty takeether
Unclassified	?	?

Table 1: Table resuming categories and topics

gories' distribution. However, the most present type is Token, which alone covers 42.4% of the dataset.

6 CONCLUSIONS

In this study, we leveraged Natural Language Techniques to understand intelligent contracts use cases. Our main contributions are the proposal of a fine-grain taxonomy of smart contracts and the presentation of insights into sub-categories with respect to previous similar works. Another contribution is that given the categories, it is possible to check quickly if a smart contract is a scam, a bank, or another type of contact. The application of LDA led us to find 15 different categories of SC that enrich those provided in previous works [2] [22] [14]. As future work, we aim to build a complete taxonomy and going deep with our analysis. In particular, we aim to find the distribution of every single topic over our dataset, taking advantage of more contracts. Also, we want to build a temporal taxonomy, showing the topics' evolution over the years and each

primary trend per year. Another contribution of the work will be creating several small datasets, each containing smart contracts of a specific topic, for example, a dataset of Non-Fungible-Tokens only.

References

1. di Angelo, M., Salzer, G.: Tokens, types, and standards: Identification and utilization in ethereum. In: 2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS). pp. 1–10 (2020). https://doi.org/10.1109/DAPPS49028.2020.00001
2. Bartoletti, M., Pompianu, L.: An empirical analysis of smart contracts: platforms, applications, and design patterns. In: International conference on financial cryptography and data security. pp. 494–509. Springer (2017)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. the Journal of machine Learning research **3**, 993–1022 (2003)
4. Bracamonte, V., Okada, H.: An exploratory study on the influence of guidelines on crowdfunding projects in the ethereum blockchain platform. In: Ciampaglia, G.L., Mashhadi, A., Yasseri, T. (eds.) Social Informatics. pp. 347–354. Springer International Publishing, Cham (2017)
5. Camino, R., Torres, C.F., State, R.: A data science approach for honeypot detection in ethereum. CoRR **abs/1910.01449** (2019), http://arxiv.org/abs/1910.01449
6. Chen, W., Zheng, Z., Ngai, E., Zheng, P., Zhou, Y.: Exploiting blockchain data to detect smart ponzi schemes on ethereum. IEEE Access **PP**, 1–1 (03 2019). https://doi.org/10.1109/ACCESS.2019.2905769
7. Dingman, W., Cohen, A., Ferrara, N., Lynch, A., Jasinski, P., Black, P.E., Deng, L.: Classification of smart contract bugs using the nist bugs framework. In: 2019 IEEE 17th International Conference on Software Engineering Research, Management and Applications (SERA). pp. 116–123 (2019). https://doi.org/10.1109/SERA.2019.8886793
8. Fenu, G., Marchesi, L., Marchesi, M., Tonelli, R.: The ico phenomenon and its relationships with ethereum smart contract environment. In: 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE). pp. 26–32 (2018). https://doi.org/10.1109/IWBOSE.2018.8327568
9. Ferreira Torres, C., Baden, M., Norvill, R., Jonker, H.: Ægis: Smart shielding of smart contracts. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 2589–2591 (2019)
10. Hu, T., Liu, X., Chen, T., Zhang, X., Huang, X., Niu, W., Lu, J., Zhou, K., Liu, Y.: Transaction-based classification and detection approach for ethereum smart contract. Information Processing and Management **58**(2), 102462 (2021). https://doi.org/https://doi.org/10.1016/j.ipm.2020.102462, https://www.sciencedirect.com/science/article/pii/S0306457320309547
11. Ibba, G., Pierro, G.A., Di Francesco, M.: Evaluating machine-learning techniques for detecting smart ponzi schemes. In: 2021 IEEE/ACM 4th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB). pp. 34–40 (2021). https://doi.org/10.1109/WETSEB52558.2021.00012
12. Jiang, B., Liu, Y., Chan, W.: Contractfuzzer: Fuzzing smart contracts for vulnerability detection. In: 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE). pp. 259–269. IEEE (2018)
13. Khan, S.N., Loukil, F., Ghedira-Guegan, C., Benkhelifa, E., Bani-Hani, A.: Blockchain smart contracts: Applications, challenges, and future trends. Peer-to-peer Networking and Applications pp. 1–25 (2021)

14. Liu, Y., Lu, Q., Xu, X., Zhu, L., Yao, H.: Applying design patterns in smart contracts. In: Chen, S., Wang, H., Zhang, L.J. (eds.) Blockchain – ICBC 2018. pp. 92–106. Springer International Publishing, Cham (2018)
15. Ortner, M., Eskandari, S.: Smart contract sanctuary https://github.com/tintinweb/smart-contract-sanctuary
16. Pierro, G.A., Tonelli, R., Marchesi, M.: Smart-corpus: an organized repository of ethereum smart contracts source code and metrics. arXiv preprint arXiv:2011.01723 (2020)
17. Shi, C., Xiang, Y., Doss, R.R.M., Yu, J., Sood, K., Gao, L.: A bytecode-based approach for smart contract classification. arXiv preprint arXiv:2106.15497 (2021)
18. Syed, S., Spruit, M.: Full-text or abstract? examining topic coherence scores using latent dirichlet allocation. In: 2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA). pp. 165–174 (2017). https://doi.org/10.1109/DSAA.2017.61
19. Tian, G., Wang, Q., Zhao, Y., Guo, L., Sun, Z., Lv, L.: Smart contract classification with a bi-lstm based approach. IEEE Access **8**, 43806–43816 (2020). https://doi.org/10.1109/ACCESS.2020.2977362
20. Wallach, H.M.: Topic modeling: Beyond bag-of-words. In: Proceedings of the 23rd International Conference on Machine Learning. p. 977–984. ICML '06, Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1143844.1143967, https://doi.org/10.1145/1143844.1143967
21. Wang, Q., Li, R., Wang, Q., Chen, S.: Non-fungible token (nft): Overview, evaluation, opportunities and challenges. arXiv preprint arXiv:2105.07447 (2021)
22. Wöhrer, M., Zdun, U.: Design patterns for smart contracts in the ethereum ecosystem. In: 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). pp. 1513–1520 (2018). https://doi.org/10.1109/Cybermatics.2018.00255
23. Zhang, P., Xiao, F., Luo, X.: A framework and dataset for bugs in ethereum smart contracts. In: 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 139–150. IEEE (2020)