What Do Programmers Discuss about Blockchain?

A Case Study on the Use of Balanced LDA and the Reference Architecture of a Domain to Capture Online Discussions about Blockchain Platforms across Stack Exchange Communities

Zhiyuan Wan, *Member, IEEE*, Xin Xia, *Member, IEEE*, and Ahmed E. Hassan, *Fellow, IEEE*

Abstract—Blockchain-related discussions have become increasingly prevalent in programming Q&A websites, such as Stack Overflow and other Stack Exchange communities. Analyzing and understanding those discussions could provide insights about the topics of interest to practitioners, and help the software development and research communities better understand the needs and challenges facing developers as they work in this new domain. Prior studies propose the use of LDA to study the Stack Exchange discussions. However, a simplistic use of LDA would capture the topics in discussions blindly without keeping in mind the variety of the dataset and domain-specific concepts. Specifically, LDA is biased towards larger sized corpora; and LDA-derived topics are not linked to higher level domain-specific concepts. We propose an approach that combines balanced LDA (which ensures that the topics are balanced across a domain) with the reference architecture of a domain to capture and compare the popularity and impact of discussion topics across the Stack Exchange communities. Popularity measures the distribution of interest in discussions, and impact gauges the trend of popularity over time. We made a number of interesting observations, including: (1) Bitcoin, Ethereum, Hyperledger Fabric and Corda are the four most commonly-discussed blockchain platforms on the Stack Exchange communities. (2) A broad range of topics are discussed across the various platforms of distinct layers in our derived reference architecture. (3) The Application layer topics exhibit the highest popularity (33.2%) and fastest growth in topic impact since November 2015. (4) The Application, API, Consensus and Network layer topics are discussed across the studied blockchain platforms, but exhibit different distributions in popularity. (5) The impact of architectural layer topics exhibits an upward trend, but is growing at different speeds across the studied blockchain platforms. The breakdown of the topic impact across the architectural layers is relatively stable over time except for the Hyperledger Fabric platform. Based on our findings, we highlighted future directions and provided recommendations for practitioners and researchers.

Index Terms—Empirical Study, Reference Architecture, Blockchain, Stack Overflow, Stack Exchange

1 Introduction

Bitcoin has emerged as the first widely-deployed, decentralized cryptocurrency. It sparked hundreds of cryptocurrencies, which in turn have attracted the attention of the financial and public regulatory sectors. The overall capitalization of cryptocurrencies has reached 305 billion USD as of Mar 2018 [24]. The core technological innovation powering cryptocurrencies is a distributed ledger known as the blockchain. Blockchain technology provides an open, decentralized and fault-tolerant transaction mechanism. Blockchain promises to become the infrastructure for a new generation of Internet interactions, including anonymous online payment [84], remittance, and transaction of digital assets [25]. Ongoing work explores smart digital contracts, enabling anonymous parties to programmatically enforce complex agreements [49], [93].

Consequently, blockchain technologies have attracted increasing interest from the development community. Blockchain-related discussions have become increasingly prevalent in programming question and answer (Q&A) websites, such as Stack Overflow and other Q&A websites in the Stack Exchange communities. Programming Q&A websites moderate hundreds of thousands of posts each month from software practitioners with a variety of backgrounds. Analyzing and understanding such knowledge repositories could provide key insights into the topics of interest to software practitioners. Prior work has conducted a wide range of empirical studies on the knowledge in programming Q&A websites [3], [9], [10], [12], [81], [88], [89], [95]. These prior studies provided insights into the categories, topics and trends in programming Q&A websites, and have uncovered programming challenges, concepts, and API usage obstacles thanks to the shared knowledge on these websites.

Most previously studied technologies via Stack Exchange communities often have long existence before Stack Exchange emerged as an important medium for developer communication and knowledge sharing. In contrast to those technologies, blockchain is a nascent technology. Thus, we

- *Zhiyuan Wan is with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China.*
E-mail: wanzhiyuan@zju.edu.cn
- *Xin Xia is with the Faculty of Information Technology, Monash University, Melbourne, Australia.*
E-mail: xin.xia@monash.edu
- *Ahmed E. Hassan is with School of Computing, Queen's University, Canada.*
E-mail: ahmed@cs.queensu.ca
- *Xin Xia is the corresponding author.*

Manuscript received ; revised

have a unique opportunity for the first time ever to spot how a development community grows its knowledge in a particular domain. We can also spot how a domain evolves in response to a variety of concerns as captured on the Stack Exchange communities. Identifying discussion topics help us pinpoint the major challenges that blockchain practitioners face.

Prior studies propose the use of latent Dirichlet allocation (LDA) topic modeling technique to study the Stack Exchange discussions (e.g., [12], [81]). In our case, the studied discussions span multiple platforms with varying levels of maturity within the same domain (i.e., blockchain). A simplistic use of LDA is not capable of dealing with our case since LDA captures the topics blindly without keeping in mind the variety of dataset and domain-specific concepts. Specifically, on the one hand, LDA is biased towards larger sized corpora as shown in prior work [46]. In our case, discussions in more active or long-lived platforms are likely to account for the dominant proportions in the dataset. On the other hand, LDA-derived topics are not linked to higher level domain-specific concepts. The derived topics are probably not suitable for the analysis of commonalities and differences across platforms.

To address the aforementioned challenges, this paper proposes an approach that combines a balanced LDA (which ensures that the topics are balanced across the domain) with the reference architecture of a domain. First, we categorized our dataset into five corpora based on blockchain platforms, and applied LDA to each categorized corpus independently. Second, we derived a reference architecture of the blockchain domain, and linked the varying concepts across the different blockchain platforms using the reference architecture. We synthesized a textual label that summarizes each topic, and mapped the labeled topics to the different layers of our derived reference architecture. Based on the architecture-mapped topics, we further analyzed the popularity and impact of topics. We also conducted a comparison across the various blockchain platforms, namely, Bitcoin [67], Ethereum [93], Hyperledger Fabric [5] and Corda [19]. The comparison provides us with an overview of cross-platform challenges and the needed support for new programming paradigms that are emerging within the blockchain domain.

Our study aims to answer the following research questions:

RQ1. What blockchain issues are being discussed?

We discovered 45 topics from the five categorized corpora in our dataset, 10 for Bitcoin corpus, 10 for Ethereum corpus, 15 for Hyperledger Fabric corpus, 5 for Corda corpus, and 5 for non-platform specific corpus. The 45 topics are further mapped to the reference architecture with 10 Application layer, 10 API layer, 1 VM Programming Language layer, 11 Consensus layer, 10 Peer-to-Peer Network layer, and 3 Developer Tools topics.

RQ2. How do discussion topics vary and evolve across different architectural layers?

The Application layer topics exhibit the highest popularity (33.2%) in blockchain discussions. The *Bitcoin: Cryptocurrency* and *Ethereum: Smart Contract* topics are most responsible for the popularity of Application layer topics. The absolute impact shows an overall increasing trend across architectural layers. The Application layer topics grew the fastest in the absolute impact since November 2015, among which, the evolution of the absolute impact of the *Bitcoin: Cryptocurrency* and *Ethereum: Smart Contract* topics is strongly associated with cryptocurrency market price. The Application layer topics have become a clear winner across architectural layers since January 2016, which attributed to the growth of Application layer topics in Ethereum Stack Exchange discussions.

RQ3. How do discussion topics vary and evolve across different blockchain platforms?

The Application, API, Consensus and Network layer topics are commonly discussed across the studied blockchain platforms, but exhibit different distributions in topic popularity. The absolute impact of the architectural layer topics exhibits an upward trend, but grows at different speeds across the studied blockchain platforms. The growth of that absolute impact mirrors the growth of cryptocurrency's market price, especially for the absolute impact of the Application layer topics. The jumps in the absolute impact coincided with fluctuation of market price, the introduction of specific Stack Exchange sites, and the news coverage of specific events. The breakdown of topic impact across the architectural layers is stable over time within each blockchain platform. For Ethereum discussions, the growth of interest slows down in the *Consensus: Mining* topic, but speeds up in the *Consensus: Transaction* and *Consensus: Address* topics. For Hyperledger Fabric discussions, an inflection point appeared in the relative impact of the *Application: Smart Contract* topic, which might be associated with its important release; The slowdown of interest in the *API: Client* topic might be associated with the maturity of documentation.

Our contributions are as follows:

1) Conceptual contribution: We derived a reference architecture along with its associated common vocabulary for the blockchain domain in an empirical data-driven fashion. The reference architecture gives software engineering researchers and others interested in blockchain a deep and empirically founded understanding of the current challenges of this new and important domain. The common vocabulary provides researchers and practitioners with a common vocabulary that captures the similarities across the various platform implementations throughout the blockchain domain.
2) Technical contribution: We proposed a novel use of Latent Dirichlet Allocation (LDA) [15] - a balanced LDA, which avoids the shortcomings of performing a simplistic LDA run to study discussions within a particular domain. Specifically, we categorized our dataset into five corpora based on blockchain platforms, built one LDA model for each corpus to capture balanced topics, and mapped the discovered topics to the various architectural layers of our derived reference architecture.
3) We performed a large-scale empirical study of blockchain-related posts across the relevant Stack Exchange communities. Specifically, we conducted quantitative and comparative analysis on blockchain-related posts across the architectural lay-

ers and across the studied blockchain platforms where appropriate, characterized the breakdown and evolution of topics, and discussed their implications on future research efforts.

The rest of the paper is structured as follows. Section 2 briefly provides blockchain background and concepts. Section 3 describes our research questions and methodology. Section 4 presents our empirical study results. Section 5 discusses the implications an threats to validity. Section 6 briefly reviews the related work. Section 7 draws conclusions and proposes avenues for future work.

2 Background

2.1 What is a blockchain?

Ledger, block and node. A blockchain is a *ledger* or, more simply, a distributed, chronological database of transactions that is shared across all the *nodes* participating in a *peer-to-peer network*. The term "blockchain" refers to transactions being grouped in blocks, and the chain of these blocks forms the accepted history of transactions. In such a blockchain, nodes update the ledger using transactions, which are signed with the private cryptographic keys of nodes.

Transaction and consensus. To ensure that only legitimate transactions are recorded into a block in the blockchain, the network confirms the validity of new transactions, given the recorded transaction history in prior blocks. Nodes interact with the blockchain via a pair of public/private keys. Nodes are addressable on the network via their public key[1]. Nodes use their private key to sign their transactions. The use of asymmetric cryptography brings authentication, integrity, and nonrepudiation to the network. Each transaction only becomes valid once it is included in a block and published to the network. The nodes on the network reach consensus on the validity of all the transactions that constitute each block. A *consensus* algorithm is used to ensure that each node in the network maintains the same copy of the blockchain as the other nodes. Then a new block of transactions is appended to the end of the blockchain. Such a process allows nodes to coordinate transactions in a decentralized manner without relying on a trusted authority to verify and clear transactions.

Cryptocurrencys and wallet. A *cryptocurrency* is a digital asset designed to work as a medium of exchange. Cryptography is used to secure the transactions and verify the transfer of assets. A cryptocurrency uses decentralized control as opposed to a central banking system. A blockchain provides a public transaction database and a distributed ledger for the decentralized control. A *wallet* stores the public and private keys that can be used to receive or spend the cryptocurrency for each client.

Smart contract. Since Nakamoto introduced Bitcoin [67], blockchain platforms now support not only the execution of cryptocurrency exchange, but also general-purpose computations on a blockchain. The *smart contract* technology is an important building block for general purpose computations on a blockchain. A smart contract is an agreement between multiple participants. The contract terms are specified using programming languages and can be executed by the blockchain nodes.

2.2 Blockchain Platforms

Ever since Bitcoin marked the introduction of blockchain platforms, a number of blockchain platforms with varied features have emerged. The emergence of various blockchain platforms enabled the creation of blockchain applications. Blockchain platforms can be divided into various categories based on two features, i.e., permissioned/permissionless and with/without smart contracts. Table 1 illustrates the characteristics and examples of each category of blockchain platforms.

Bitcoin [67] is the first widely-deployed blockchain platform, launched in August 2008. *Ethereum* [93] is an open-source blockchain platform, proposed in late 2013. Ethereum features smart contract functionality. *Hyperledger*[2] was started in December 2015 by the Linux Foundation. It tends to support the collaborative development of blockchain platforms and applications. Within various projects in Hyperledger, *Hyperledger Fabric* [20], first released in February 2016 (v0.1.0)[3], is a modular blockchain platform that is designed to support pluggable components. *Hyperledger Composer*[4] is comprised of a development toolset and a framework to integrate blockchain applications with existing blockchain platforms. Corda records, manages and automates financial and legal agreements between business partners [19]. Corda's code was open-sourced in November 2016.

2.3 Reference Architectures

Reference architectures have emerged as abstractions of the software architectures of systems within specific domains [6]. The reference architecture of a given domain defines fundamental components of the domain and the relations between these components [38]. Reference architectures bring a number of benefits. First, a reference architecture establishes common mechanisms to improve the interoperability among different software systems and their components [23]. Reference architectures often act as a knowledge repository to improve communication and information exchange [60]. A reference architecture provides generic artifacts, architectural styles, and domain vocabulary for software systems in a particular domain [30]. As new products are developed in a new domain, the designers develop new sets of concepts and terminology. The lack of consistency among concepts and terminology of a domain often hinders comparisons and interoperability in a new domain. A reference architecture provides a common vocabulary to communicate and discuss challenges across the software systems in the same domain [38], thus facilitates the uniform description of architectures for those software systems, and further helping a community grow and mature. A reference architecture gives software engineering researchers and others interested in a particular domain a

[1] Depending on the implementation, the address can be the public key itself or (usually) a hash of it.

[2] https://www.hyperledger.org

[3] https://github.com/hyperledger-archives/fabric/releases

[4] https://hyperledger.github.io/composer

TABLE 1
Categories of blockchain platforms.

Category	Characteristic	Example
Permissionless	• fully public chains • maintained by public nodes • accessible to anyone	Bitcoin, Ethereum
Permissioned	• define different permissions on different nodes • only involve authorized nodes • facilitate faster, more secure and cost-effective transactions	Hyperledger Fabric, Corda
With smart contract	• enable smart contract capabilities • facilitate building business logic into chains	Ethereum, Hyperledger Fabric, Corda
Without smart contract	• only built for transaction capabilities	Bitcoin

deeper and empirically founded understanding of the current structure and challenges of that new domain. Previous studies show that reference architectures enable the reuse of common assets, and thus moderate the development costs of software projects and lower time-to-market of the constructed software [23], [59].

Prominent examples of reference architectures include the reference architecture for web services [37], AUTOSAR [52] for automotive software, a set of references architectures presented in the Microsoft Application Architecture Guide [65] supported by the Microsoft technology stack, NIST cloud computing reference architecture [55], IBM's big data reference architecture [11], Oracle's reference architecture for big data systems [21], vendor-independent reference architectures for the Internet of Things [92] and OATH for authentication [74]. Software reference architectures come in different flavors depending on their context of use (e.g., used within or across organizations), origin (third parties or designed in-house) and purpose (e.g., facilitating development and standardization) [6]. Reference architectures have been used in the past as part of the commonly used ATAM approach for evaluating architectural alternatives [45]. Reference architectures in conjunction with ATAM have been used as well to enable more structured and guided comparisons of software solutions as part of large scale software acquisitions [29]. In addition, some of the previously published reference architectures in academic venues represent well-known technological domains, including web servers [38], web browsers [36], and software testing tools [75].

2.4 Topic Modeling

Topic modeling is an advanced statistical technique that can automatically discover *topics* from a given text corpus, without the need for tags, training data, or predefined taxonomies [14], [35]. Topic modeling uses word frequencies and co-occurrence frequencies in the documents to build a model of related words [12]. Topic modeling has been successfully used in other tasks to automatically analyze millions of unstructured documents, including analyzing news articles [70], extracting topics from bug reports [56], and identifying topics in the source code [51].

The latent Dirichlet allocation is a powerful topic modeling technique, which has previously been used for automated text mining and clustering of the Stack Exchange posts [12], [81], [95], and enables the qualitative analysis to gain insights into characteristics and trends in the dataset. A recent study [62] also leveraged LDA as a text mining approach to find the trends in software research. LDA represents topics as probability distributions over the words in the corpus and documents as probability distributions over the discovered topics. LDA discovers topics based on how sets of words tend to co-occur frequently in the documents of the corpus. The words in a discovered topic are semantically related and give meaning to the topic as a whole. Each document of the corpus contains a mixture of topics. Topics exist across multiple documents, making it possible for LDA models to discover granular themes that represent the corpus as a whole.

Prior work [87] shows that LDA is biased towards corpus with a larger size. When an approach feeds LDA multiple corpora at once, the resulting topics are primarily derived from the larger corpus, since larger corpus dominates the corpora in sheer size. Kelly et al. [46] referred to this approach as "unbalanced" approach.

3 Research Settings

3.1 Research Questions

RQ1. What blockchain issues are being discussed?

Programming Q&A websites are designed to cater to the needs of practitioners facing challenges. Identifying the discussion topics can help us pinpoint the major challenges that blockchain practitioners are currently facing. Software engineering researchers could objectively discover real issues. Accordingly, future research efforts can focus on these relevant issues to help support and improve the quality of blockchain platforms.

RQ2. How do blockchain topics vary and evolve across different architectural layers?

Previously studied technologies through the mining of data from Stack Exchange communities had usually existed before Stack Exchange emerged as an important medium for developer communication and knowledge sharing. In contrast, blockchain is a nascent technology. We have a unique opportunity for the first time ever to spot how a development community grows and shares its knowledge in a particular domain, as well as how a domain evolves in response to a variety of concerns as captured through the discussions on the Stack Exchange communities.

In addition, the Bitcoin and Ethereum communities on Stack Exchange are technical communities associated with the cryptocurrencies. Thus, we have a unique opportunity for the first time ever to spot how the technical communities on Stack Exchange have evolved in response to the real-life dynamics of cryptocurrencies.

RQ3. How do blockchain topics vary and evolve across different blockchain platforms?

Different blockchain platforms have distinct inherent features. For instance, Blockchain 1.0 describes cryptocurrencies like Bitcoin, enabling the transactions of digital property; Blockchain 2.0 describes more complex interactions, the creation of new decentralized economies and financial instruments, based upon "smart contracts", as provided by Ethereum [85]. Do the inherent features of blockchain platforms lead to differences in discussed topics across the blockchain platforms? Understanding how blockchain topics vary and evolve gives insights to the potential of blockchain platforms, and serves as guidance cues that may help a community grow and mature.

3.2 Research Methodology

Our research methodology consists of the following five steps which are detailed in the following subsections.

3.2.1 Collecting Data

Stack Overflow[5] is a popular programming Q&A website where knowledge related to programming and software engineering tasks is exchanged. Stack Overflow enables users to ask new questions and answer existing questions in a variety of areas. Users of Stack Overflow can "vote" questions and answers up or down, based on the perceived value of each post, as well as earn reputation points and "badges" through various activities.

Stack Exchange[6] launched in 2008 with Stack Overflow as its first Q&A site for programming and software engineering questions. Stack Exchange continues to grow and adds more specific Q&A sites that cover diverse topics, including:

- 70 *Technical* communities, e.g., Ubuntu, Server Fault, and Information Security.
- 46 *Culture and Recreation* communities, e.g., English Language & Usage, Travel, and Bicycles.
- 24 *Life and Arts* communities, e.g., Science Fiction & Fantasy, Photography, and Law.
- 19 *Science* communities, e.g., Mathematics, Computer Science and Philosophy.
- 7 *Professional* communities, e.g., The Workplace, Open Source, and Writing.
- 3 *Business* communities, e.g., Quantitative Finance, Project Management, and Patents.

Stack Exchange makes its data publicly available in XML format under the Creative Commons License[7]. The data dump is divided into five XML documents: `Posts.xml`, `Comments.xml`, `Badges.xml`, `Users.xml` and `Votes.xml`. For our purposes, we use the `Posts.xml`, which contains the actual text content of all posts, as well as a unique ID for each post, along with its creation date, its type (question or answer) and its associated tags. Our studied Stack Overflow dataset spans 9 years and 4 months, from July 2008 to December 2017 and contains 38,485,045 posts in total.

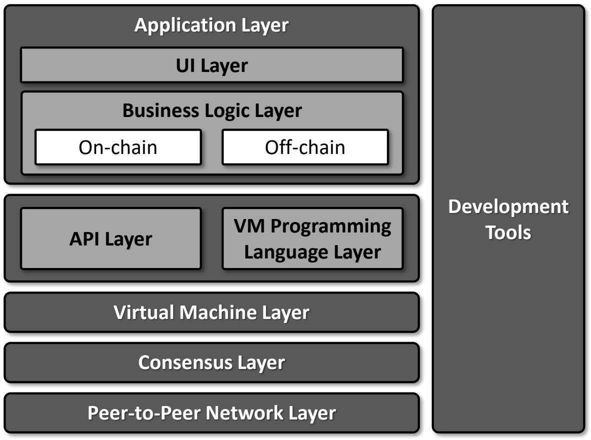


Fig. 1. Reference architecture of blockchain platforms.

Since the posts on Stack Overflow can be about any topic, we followed a process similar to prior work [54], [81] and tried to identify blockchain-related posts. See Appendix A for more details about how we collected our data.

3.2.2 Deriving a Reference Architecture

We followed a similar process as discussed in [36], [38] and derived a reference architecture for the blockchain domain. The process for deriving the reference architecture consists of the following steps:

Step 1: Derive conceptual architectures. First, we proposed conceptual architectures for the studied blockchain platforms, using domain knowledge and available documentation. Then we recovered the concrete architectures from the source code. Finally, we refined the conceptual architectures using the recovered concrete architectures.

Step 2: Derive a reference architecture. We compared the refined conceptual architectures of the studied blockchain platforms to find common components and relations between components. Based on the comparison of conceptual architectures and the architecture documentation of various blockchain platforms [2], [18], [64], [94], we inferred a reference architecture for the blockchain domain.

The reference architecture we derived consists of five layers of capabilities as depicted in Fig. 1. We describe each layer in the five-tier architecture as follows.

1) **Peer-to-peer network layer** is responsible for inter-node communications, including node discovery, transactions and block propagation.
2) **Consensus layer** is one of the most crucial layers in all blockchain platforms. The layer provides implementations to generate the order of block creations and validate blocks created by other nodes in the network.

[5] https://stackoverflow.com
[6] https://stackexchange.com
[7] https://archive.org/details/stackexchange

3) **Virtual machine layer** is a "transaction engine" present in all blockchain platforms. The layer is responsible for changes in a blockchain's global state.
4) The fourth blockchain architectural layer consists of two sub-layers: 4.a **API layer** is represented by interfaces in blockchain nodes and used by blockchain applications at runtime; 4.b **VM programming language layer** is used at development time and compiled into virtual machine code that can be deployed into a blockchain and executed by the virtual machine layer.
5) **Application layer** consists of two sub-layers: **business logic layer** and **UI layer**. The business logic layer provides application-specific functionalities by using the underlying blockchain in order to deliver business solutions. *Smart contract*s reside in the business logic layer.
   The business logic layer consists of *On-chain* and *Off-chain* parts. The **on-chain** part is the actual smart contract that is usually written in some VM programming language (Layer 4), executed by a particular virtual machine (Layer 3), and becomes part of the blockchain state (Layer 2). The **off-chain** part links the smart contract business logic to the outside world.
   The UI layer is the interface of the blockchain applications as presented to its users. This layer calls the underlying functionalities in the business logic layer.

3.2.3 Identifying Topics

Our research goals involve identifying the discussion topics in blockchain platforms. We used user-specified tags as a starting point to discover general discussion topics. However, these tags fail to provide fine-grained categorization information. Instead, they simply provide us with a general sense of a question. For example, knowing that a question has a "bitcoin" tag fails to tell us whether it is a wallet related issue or a cryptocurrency related issue. The drawback of tags motivates us to consider a different approach to discover the discussion topics in blockchain related discussions. As a result, we use *LDA* to capture fine-grained discussion topics in our dataset.

LDA implementation. We use the implementation of LDA in the gensim Python library[8], which implements online LDA [42] - an online variational Bayes (VB) algorithm for LDA. We notice that prior work [77] proposed generic algorithms based approach to determine LDA parameters when applying LDA to the source code. They claimed that source code is much more repetitive and predictable as compared to natural language; thus, using parameters for natural language texts did not always produce expected results. However, our corpora only include natural language texts. Thus, we run gensim LDA for five passes with K topics and default settings for all the other parameters.

Number of topics. The number of topics of LDA, denoted K, is a user-specified parameter that influences the performance and provides control over the granularity of the discovered topics [12]. Coherence measures are proposed to rate the understandability of the discovered topics by topic models [71]. To find the optimal K values to derive topics with high coherence, we experimented with different values of K to capture topics with the highest coherence measures. We used coherence measures to quantify the coherence of a fact set [17]. Specifically, we the set K of our LDA models to range between 5 and 50 with a step of 5 respectively. We used the gensim module `CoherenceModel` to calculate the topic coherence for each computed LDA model (i.e., 5, 10, 15, ..., 50). The gensim module `CoherenceModel` implements the Röder et al.'s four-stage topic coherence pipeline [80].

Option A: single LDA run for all corpora. We started by building a single topic model for our studied dataset. We ran LDA with different topic numbers, and found that the topic models with 10 topics achieve the optimal topic coherence. The 10 discovered topics include *Wallet*, *Hyperledger Fabric*, *Smart Contract*, *Transaction*, *Client*, *API*, *Consensus Protocol*, *Cryptocurrency*, *Corda*, and *npm*. We notice that discovered these topics are of different levels of granularity and at different layers of our proposed reference architecture: (1) The *Hyperledger Fabric* and *Corda* topics are blockchain platforms; (2) The *Wallet*, *Smart Contract*, and *Cryptocurrency* topics reside at the application layer of the reference architecture; (3) The *API* and *Client* topics are at the API layer of the reference architecture; (4) The *Transaction*, *Client*, *Consensus Protocol* topics discuss important building blocks of the consensus layer; (5) The *npm* topic discusses a package manager for JavaScript, which is a development tool.

As shown in prior work [46], LDA is biased towards larger sized corpora. In our case, the resulting topics are primarily derived from the Bitcoin and Ethereum platforms. This is because Bitcoin corpus and Ethereum corpus dominate the dataset in sheer size over other corpora. The discovered topics are not suitable for further analysis of commonalities and differences across blockchain platforms.

Option B: one LDA run per corpus. We leveraged user-specified tags to categorize the posts in our Stack Overflow dataset based on blockchain platforms. First, we identified the tags co-occurring with top-ranked platform tags, i.e., "bitcoin", "ethereum", "hyperledger" and "corda", respectively. The co-occurring tags with each platform tag became one group of tags. Then, we recognized exclusive tags in each group. The lists of exclusive tags for each group are shown in Table 2. Finally, we categorized the posts in our Stack Overflow dataset using these lists. As a result, our Stack Overflow dataset breaks into five corpora: 1,158 posts related to Bitcoin, 821 posts related to Ethereum, 1,328 posts related to Hyperledger Fabric, 190 posts related to Corda, and other 401 non-platform specific posts.

We further combined the Stack Overflow dataset with the Stack Exchange dataset and obtained five corpora: 18,092 posts in Bitcoin corpus, 12,792 posts in Ethereum corpus, 1,328 posts in Hyperledger Fabric corpus, 190 posts in Corda corpus, and 401 posts in non-platform specific corpus. We then executed one LDA run per corpus to capture topics for each corpus in our dataset. This process allowed us to use a different number of K topics for each corpus based on its complexity and richness.

[8] https://radimrehurek.com/gensim/

TABLE 2
List of tags used to categorize blockchain-related posts on Stack Overflow.

Bitcoin	Ethereum	Hyperledger	Corda
bitcoin	ethereum	hyperledger	corda
bitcoind	solidity	hyperledger-fabric	
bitcoinj	smartcontracts	hyperledger-composer	
bitcoin-testnet	truffle	hyperledger-explorer	
nbitcoin	web3js		
bitcoinlib			

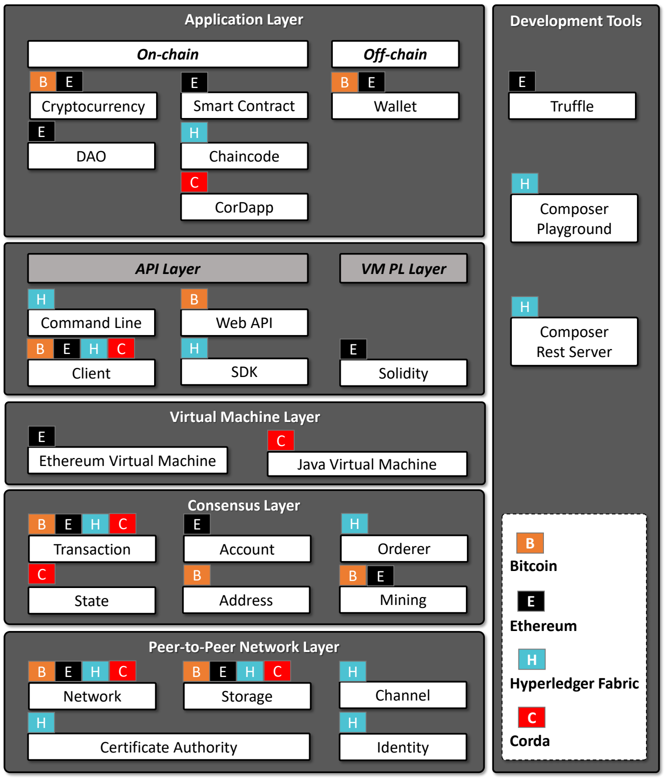


Fig. 2. Mapping of the studied Blockchain Platforms to the Reference Architecture of the Blockchain Domain.

3.2.4 Mapping Topics Using the Reference Architecture

Different blockchain platforms provide various specific implementations for the aforementioned layers in the reference architecture. To enable us to study various platforms simultaneously, we mapped the concepts in different blockchain platforms to the different layers of our derived reference architecture. The architecture-mapped concepts are depicted in Fig. 2.

Application layer. Bitcoin and Ethereum platforms both have *Cryptocurrency* as their on-chain application layer and wallets as their off-chain application layer. *Chaincode* and *CorDapp* are the specific implementations of smart contracts on the Hyperledger Fabric and Corda platforms respectively.

API layer. Bitcoin, Ethereum, Hyperledger Fabric and Corda platforms provide specific *clients* to access blockchains, e.g., *bitcoind* in Bitcoin and *go-ethereum* (*geth*) in Ethereum. The Bitcoin platform also provides Web APIs, e.g, *blockchain.info-api*. In addition, the Hyperledger Fabric platform provides a command-line interface and SDKs in multiple languages (e.g., `fabric-sdk-node` for Node.js and `fabric-sdk-java` for Java).

VM programming language layer. *Solidity*[9] is a contract-oriented, high-level language for implementing smart contracts on the Ethereum platform. It is designed for the Ethereum Virtual Machine (EVM).

Virtual machine layer. Ethereum smart contracts run in the EVM, which handles states and computation on the Ethereum platform. CordApps run in the Java Virtual Machine which handles states and computations on the Corda platform.

Consensus layer. The four studied platforms use two different transaction models to keep track of transactions in their consensus layer, i.e., UTXO (unspent transaction output) model[10] and account model. In a UTXO model, transactions are linked to each other in a consumer-producer model: Each transaction takes current entries in the ledger as input to produce new entries as output. In an account model, stateful accounts are maintained on the ledger: each transaction takes the form of requests to update the current state of these accounts. The Bitcoin and Corda platforms use a UTXO model, while the Ethereum and Hyperledger platforms use an account model. A Bitcoin *address* is a unique identifier that represents the origin or the destination of a transaction. An Ethereum *Account* represents the identity of an external agent (e.g., human personas, mining nodes or automated agents), which has an account balance (and contract storage if it is a contract account). A Corda *State* is an immutable object representing a fact of any kind (e.g., stocks, bonds, and loans) at a specific point in time. A Hyperledger Fabric *Orderer* provides an ordering service that orders transactions in a block. Mining is the process of coming into consensus about the validity of transactions.

Peer-to-peer network layer. Each blockchain platform manages a distributed ledger using a peer-to-peer *network*. The network adheres to a protocol of inter-node communication. The protocol enables the validation of new blocks. Nodes in the network of different blockchain platforms leverage different solutions for data *storage*, e.g., Hyperledger Fabric uses `CouchDB` for the state database.

In the Hyperledger Fabric platform, each actor (nodes and applications) in the network has a digital *identity* that is encapsulated within an X.509 digital certificate. *Identities* determine the exact permissions over resources and access to information that actors have in the network. The certificates are issued by Certificate Authority component - *Hyperledger Fabric CA*. The *Channel* is a mechanism by which actors within a blockchain network can communicate and transact privately. Actors in a channel come together to collectively share and manage identical copies of the ledger for that channel.

Development tools. *Truffle* is an Ethereum development framework that helps developers debug, deploy, and test smart contracts. Hyperledger Composer *Playground* provides a user interface for the configuration, deployment, and testing of Hyperledger Fabric business networks. The Hyperledger Composer *REST Server*[11] can be used to gen-

[9] https://github.com/ethereum/solidity
[10] https://docs.corda.net
[11] https://hyperledger.github.io/composer/latest/reference/rest-server

erate a REST API from a deployed business network. The REST API can be easily consumed by HTTP or REST clients.

To clarify the semantics of the topics and ease readability, we synthesized a textual label that summarizes each topic. Specifically, we examined the top words in a topic and reviewed a random sample of 10 questions whose dominant topics are assigned to that topic. We further validated the topic names with two domain experts. By comparing top words of each topic with the architecture-mapped concepts (see Fig. 2), we map the labeled topics into layers of our derived reference architecture. We conduct further analysis based on the architecture-mapped topics.

3.2.5 Metrics and Analysis

We apply LDA to each corpus in our dataset, obtaining a set of K *topics* $(z_1, ..., z_k)$ and a set of *topic probability vectors*. Topics are distributions over the unique words in each corpus; the topic probability vector of each post indicates the proportion of words that come from each of the K topics. We denote the probability of a particular topic z_k in post d_i as $\theta(d_i, z_k)$. Note that $\forall i, k : 0 \leq \theta(d_i, z_k) \leq 1$ and $\forall i : \sum_1^k \theta(d_i, z_k) = 1$.

Dominant topic.

As used in [77], [81], we define the *dominant topic* for each post as the topic with the highest topic probability. The concept of a dominant topic allows us to cluster the posts based on their *topic probability vectors*. Formally, the dominant topic of a post d_i is defined as

$$dominant(d_i) = z_k : \theta(d_i, z_k) = max(\theta(d_i, z_j)), 1 \leq j \leq K \tag{1}$$

Topic popularity. We define topic popularity for each topic z_k within a corpus c_j as

$$popularity(z_k, c_j) = \frac{|\{d_i\}|}{|c_j|} : dominant(d_i) = z_k, 1 \leq i \leq c_j \tag{2}$$

We define topic popularity for an architecture-mapped topic M within a corpus c_j as

$$popularity(M, c_j) = \sum_{z_k \in M} popularity(z_k, c_j) \tag{3}$$

We define topic popularity for the topics at an architecture layer L within a corpus c_j as

$$popularity(L, c_j) = \sum_{z_k \in L} popularity(z_k, c_j) \tag{4}$$

Topic trends over time. To analyze the temporal trends of topics, we define the absolute *impact* metric of a topic z_k in month m as

$$impact_{absolute}(z_k, m) = \sum_{d_i \in D(m)} \theta(d_i, z_k) \tag{5}$$

where $D(m)$ is the set of posts in month m. The absolute *impact* metric measures the absolute proportion of posts related to a particular topic z_k in a particular month m.

We further define the absolute *impact* metric of topics at a particular architectural layer L in month m as

$$impact_{absolute}(L, m) = \sum_{z_k \in L} impact_{absolute}(z_k, m) \tag{6}$$

We define the relative *impact* metric of topic z_k in month m within corpus c as defined in [12]:

$$impact_{relative}(c, z_k, m) = \frac{1}{|D(c, m)|} \sum_{d_i \in D(c,m)} \theta(d_i, z_k) \tag{7}$$

where $D(c, m)$ is the set of posts in month m within the corpus c where z_k is discovered. The relative *impact* metric measures the relative proportion of posts related to a particular topic z_k in a particular month m.

We define the relative *impact* metric of architecture-mapped topic a_k in month m within corpus c as

$$impact_{relative}(c, a_k, m) = \sum_{z_k \in a_k} impact_{relative}(c, z_k, m) \tag{8}$$

We define the relative *impact* metric of topics at a particular architectural layer L in month m within corpus c as

$$impact_{relative}(c, L, m) = \sum_{z_k \in L} impact_{relative}(c, z_k, m) \tag{9}$$

where $D(m)$ is the set of posts in month m.

4 Results

We now present the results of applying our research methodology on our dataset. We succinctly answer each of our research questions.

4.1 RQ1. What blockchain issues are being discussed?

Bitcoin, Ethereum, Hyperledger Fabric and Corda are the four most commonly-discussed blockchain platforms on the Stack Exchange communities. In addition, the Stack Exchange communities discussed about other blockchain technologies, e.g., IOTA[12], Dogecoin[13], and Eris[14]. The first post on Stack Exchange appears in May 2011 for Bitcoin, May 2015 for Ethereum, March 2016 for Hyperledger Fabric, and July 2016 for Corda, respectively.

LDA discovered topics. We discover 45 topics from the five corpora in our dataset, 10 for Bitcoin corpus, 10 for Ethereum corpus, 15 for Hyperledger Fabric corpus, 5 for Corda corpus, and 5 for non-platform specific corpus. The details of the discovered topics and their top key words are shown in Table 8 (see Appendix C). We then identify the dominant topic for each post by using the *dominant topic* metric (1). We classify the posts according to their dominant topic, and measure *popularity* (2) for each topic. The topics within each corpus are arranged in descending order according to the topic *popularity*. Table 9 in Appendix C shows the trendline of each topic in the third column, using the relative topic *impact* metric (7). These trendlines give us an indication of the rise or fall of interest in a particular topic.

Architecture-mapped topics. Table 3 present the architecture-mapped topics discovered by our methodology. The first column shows the architecture-mapped topics. From the second to the sixth column, the length of each rectangle indicates the proportion of posts in a particular

[12] https://www.iota.org
[13] https://dogecoin.com
[14] https://erisindustries.com

TABLE 3
The architecture-mapped topics in the five studies corpora. "× number" indicates the number of low-level topics that are mapped into a particular architecture-mapped topic. The length of rectangles indicates the percentage of posts in a particular topic within a corpus (relative to the column).

Topics	Bitcoin	Ethereum	Hyperledger Fabric	Corda	Non-Platform Specific
Application Layer					
Cryptocurrency	×1	×1			
Wallet	×2	×1			
Smart Contract		×1	×2	×1	
DAO		×1			
API Layer					
Client	×1	×1	×2	×1	
Web API	×1				
Java SDK			×1		
Library					×3
VM PL Layer					
Solidity		×1			
Consensus Layer					
Transaction	×1	×1	×1	×1	×2
Address	×1				
Mining	×1				
Account		×1			
Orderer			×1		
State				×1	
Network Layer					
Network	×1		×1		
Storage	×1	×1	×1	×1	
Channel			×2		
Fabric CA			×1		
Identity			×1		
Development Tools					
Truffle		×1			
Playground			×1		
REST Server			×1		

topic within a corpus (relative to column). "× number" indicates the number of low-level topics that are mapped into a particular architecture-mapped topic. We present a subset of representative example posts for each topic in Appendix D. For the rest of our analysis, we focus our discussions on architectural layer or architecture-mapped topics instead of examining each specific raw topic.

We discover 45 topics from the five corpora in our dataset, 10 for Bitcoin corpus, 10 for Ethereum corpus, 15 for Hyperledger Fabric corpus, 5 for Corda corpus, and 5 for non-platform specific corpus.

Out of the 45 topics, the Application layer, API layer, VM Programming Language layer, Consensus layer, Peer-to-Peer Network layer, and Developer Tools topics account for 10, 10, 1, 11, 10, 3 topics, respectively.

4.2 RQ2. How do discussion topics vary across different architectural layers?

4.2.1 Topic Popularity

We compared the popularity of topics at each architectural layer for our studied corpora. The barplot in Fig. 3 shows the topic popularity at various architectural layers. Discussions of the Application layer topics have the highest topic popularity (33.2%), followed by the discussions of the Consensus layer topics (26.5%). The top 2 contributors to the popularity of the Application layer topics are the *Cryptocurrency* topic (9.2%) for Bitcoin and the *Smart Contract* topic for Ethereum (7.2%).

The Application layer topics exhibit the highest popularity (33.2%) in blockchain discussions. The *Bitcoin: Cryptocurrency* and *Ethereum: Smart Contract* topics are most responsible for the popularity of Application layer topics.

4.2.2 Topic Absolute Impact

We investigated the trends of absolute *impact* of the topics at various architectural layers. We calculated the absolute *impact* scores (6) for the topics at different architectural layers between the time period of their first appearance and the time period of their last appearance. Fig. 3 plots the absolute topic impact of various architectural layers from May 2011 and November 2017. We observed an overall increase in the absolute topic impact for all the architectural layer topics. The significant increase in absolute topic impact indicates the growth of the blockchain communities on Stack Exchange, with no signs of a "tipping point" in the growth.

Among the topics of various layers, the Application layer topics are most responsible for the increasing trend in absolute impact over time, especially its intensive increase in absolute impact since November 2015. We further plotted the absolute impact scores of the top contributors in application layer - the *Cryptocurrency* topic in Bitcoin (Fig. 5(a)) and the *Smart Contract* topic in Ethereum (Fig. 5(b)), and the market prices of Bitcoin and Ethereum. We computed the Pearson's product-moment correlation coefficients between the absolute impact of the *Cryptocurrency* topic in Bitcoin and Bitcoin market prices (cor = 0.76, p-value<7.2e-16), and between absolute impact of the *Smart Contract* topic in Ethereum and Ethereum market prices (cor = 0.92, p-value<2.5e-12). The absolute impact of top-contributor top-

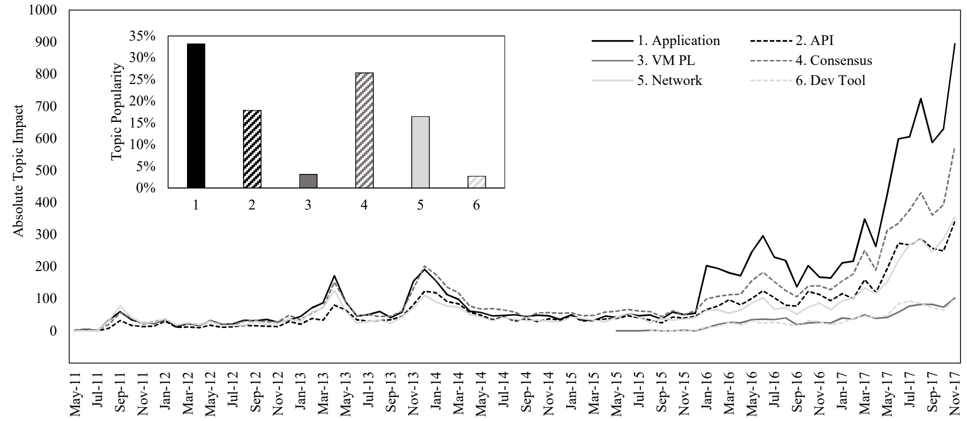


Fig. 3. The *popularity* and absolute *impact* scores of the topics at various architectural layers in our dataset between May 2011 and November 2017.

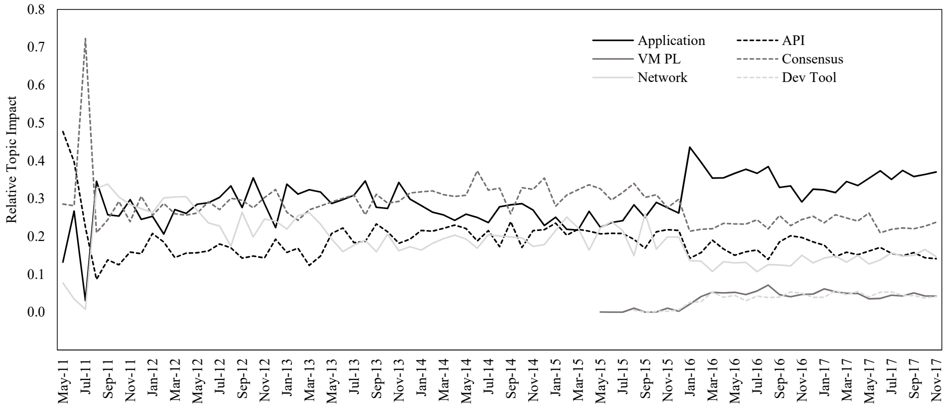


Fig. 4. The relative *impact* scores of topics at various architectural layers in our dataset between August 2015 and November 2017.

ics is strongly associated with the market price. The rising market price may provide a strong incentive to the growth of the blockchain communities on Stack Exchange.

Interestingly, the absolute impact of top contributors at the Application layer over time is likely to mirror the evolution of the blockchain technology [85]. Specifically, *Blockchain 1.0* describes cryptocurrencies like Bitcoin, enabling the transactions of digital property; *Blockchain 2.0* describes more complex interaction, the creation of new decentralized economies and financial instruments, based upon "smart contracts", as offered by Ethereum.

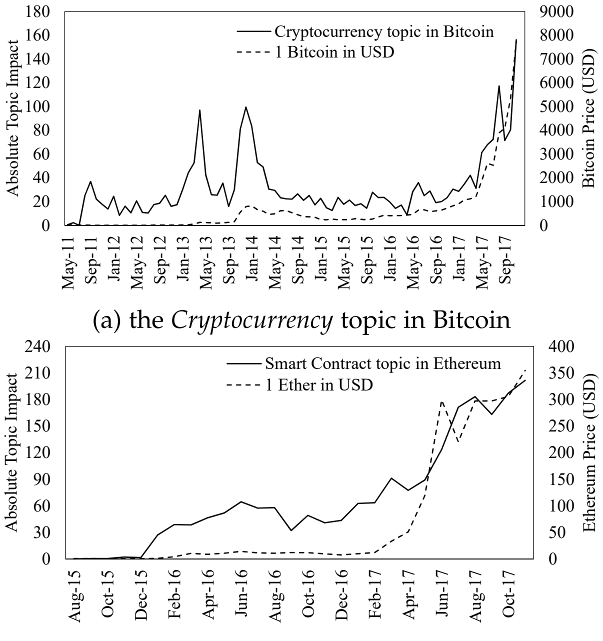
The absolute impact shows an overall increasing trend across architectural layers. The Application layer topics grow fastest in absolute impact since November 2015. Among Application layer topics, the evolution of absolute impact of the *Bitcoin: Cryptocurrency* and *Ethereum: Smart Contract* topics is strongly associated with cryptocurrency market price.

4.2.3 Topic Relative Impact

We measure the relative *impact* scores (9) for each of the various architectural layers for our studied corpora. Fig. 4 plots the relative impact scores of topics at each architectural layer. The relative impact scores indicate the evolution of relative proportions of the topics at each architectural layer.

We observe that the topics at the Application and Consensus layers showed comparable relative impact before January 2016. The increase in the relative impact of the Application layer topics is likely associated with the introduction of Ethereum Stack Exchange site in August 2015. From August 2015 to January 2016, the Consensus layer and application layer topics for Ethereum have undergone extensive changes in relative impact: the Consensus layer topics have experienced a sharp decrease, and later exhibited a relatively flat trend; Meanwhile, the Application topics for Ethereum have undergone an intense increase. Since January 2016, the Application layer topics are the key drivers of the relative impact across architectural layers. The Ethereum community on Stack Exchange appears to be more concerned about Application layer than other layers.

We further checked whether the relative impact of topics at each architectural layer is increasing or decreasing over time to a statistically significant degree using the Cox Stuart trend test [26]. The Application layer (p-value = 0.0047) and Development Tools topics (p-value = 0.0065) have a statistically significant increasing trend; The Consensus (p-value = 0.027) and Network layer topics (p-value = 1.7e-07) have a statistically significant decreasing trend; The API and VM Programming Language layer topics have a constant trend.



(a) the *Cryptocurrency* topic in Bitcoin

(b) the *Smart Contract* topic in Ethereum

Fig. 5. The absolute *impact* scores of top contributors among application layer topics, as compared with the trends of market prices of Bitcoin and Ethereum.

The Application layer topics have become a clear winner across architectural layers since January 2016, which attributed to the growth of Application layer topics in Ethereum discussions on Stack Exchange.

4.3 RQ3. How do discussion topics vary across different blockchain platforms?

RQ3 considers the topic popularity and topic evolution across different blockchain platforms. The comparison across blockchain platforms gives us valuable insights about the similarities and differences in topics of various blockchain platforms. We first measured the topic popularity for the topics at various architectural layers across the studied blockchain platforms. Fig. 6 plots the topic popularity at various architecture layers (4) for each of the studied blockchain platforms. We then measured the trends in the absolute topic impact of topics across the studied blockchain platforms. Finally, we compared the trends in the relative topic impact of the architecture-mapped topics across the studied blockchain platforms.

4.3.1 Topic Popularity

The Application layer topics are discussed across the studied blockchain platforms. Besides the absence of smart contracts in the Bitcoin platform, the *Smart Contract* topic is popular across our studied blockchain platforms. Notice that different blockchain platforms support different smart contracts, i.e., *Smart Contract* for Ethereum, *Chaincode* for Hyperledger Fabric, *CorDapp* for Corda. In contrast to traditional distributed applications, smart contracts are irreversible and immutable, i.e., they cannot be patched when bugs are detected. There is no way to fix a buggy smart contract without reversing the blockchain regardless of its popularity. Meanwhile, bugs in a smart contract could bring disastrous damages. It is critical to thoroughly test smart contracts and reason about the correctness of smart contracts before their deployment. High-level programming languages (e.g., Solidity and Serpent) have been designed to facilitate the development of smart contracts. Future research could take into consideration bug detection and localization techniques for smart contract oriented programming languages.

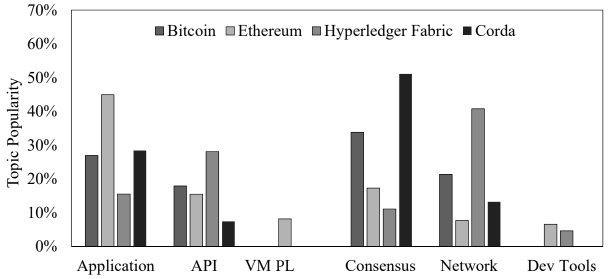


Fig. 6. The topic *popularity* at various architectural layers across the studied blockchain platforms.

The API layer topics are discussed across the studied blockchain platforms. Among application layer topics, the *Client* topic is popular across all studied blockchain platforms. Note that different blockchain platforms provide implementations of clients according to distinct protocols, e.g., *bitcoind* in Bitcoin and *geth* in Ethereum. *bitcoind* has been bundled with original Bitcoin client (the *Satoshi client*) from version 0.2.6 to 0.4.9, and with *Bitcoin Core* since version 0.5.0. From January 2009 to November 2017, *bitcoind* has in total 84 releases and the latest release is version 0.15.1[15]. From February 2014 to November 2017, *geth* has in total 105 releases and the latest release is version 1.7.3[16].

The Consensus layer topics are discussed across the studied blockchain platforms. Among Consensus layer topics, the *Transaction* topic is popular across all studied blockchain platforms. Questions related to the *Transaction* topic discuss building, sending, analyzing and validating transactions, cryptographic primitives in transactions, transaction fees, etc. Failures in transactions intensively hurt the reliability and usability of blockchain platforms. To improve the reliability and usability of blockchain platforms, it is essential to develop transaction-aware software engineering tools (e.g., transaction-level debugger, logger and testing tool) to help practitioners manipulate transactions throughout the transaction life cycle.

The Network layer topics are discussed across the studied blockchain platforms. Among Network layer topics, the *Storage* topic is popular across all studied blockchain platforms. The *Storage* topics concerned about "storing" data for the blockchain platforms, including the ledger and meta data (e.g., state). Bitcoin and Hyperledger Fabric separately store the data: ledger into raw files and meta data into databases; while Ethereum and Corda store all the data into databases. Different platforms provide various options for database storage, e.g., LevelDB, CouchDB, RocksDB, H2,

[15] https://en.bitcoin.it/wiki/Bitcoind

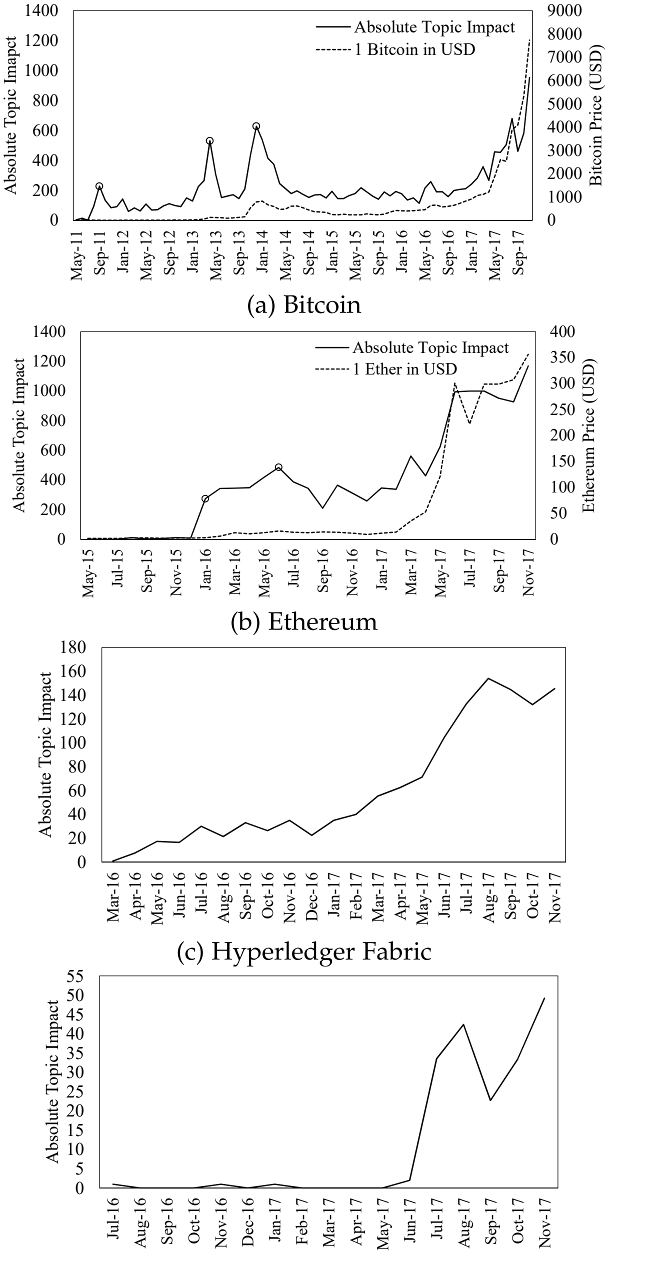
[16] https://github.com/ethereum/go-ethereum/releases

PostgreSQL, and SQLServer[17] [18] [19] [20].

The distributions of topic popularity at various architectural layers differ across blockchain platforms. As the representative platform of Blockchain 1.0, Bitcoin has a relatively balanced distribution. Other platforms have relatively unbalanced distributions of topic popularity at various layers. As the representative platform of Blockchain 2.0, Ethereum discussions have the greatest popularity for the Application layer topics. As the innovative technology of Blockchain 2.0, the *Smart Contract* topic at the Application layer has attracted most of the interest in Ethereum. As the successors of Blockchain 2.0 platforms, Hyperledger Fabric discussions have the greatest popularity in the topics at the Network layer. The topics with greatest popularity are at the Consensus layer for Corda.

The topics at the VM Programming layer are only popular in Ethereum, the topics at the Development Tool layer are only popular in Ethereum and Hyperledger Fabric. The VM programming language layer topics are only popular in Ethereum because only the Ethereum platform has its own programming languages (e.g., Solidity, Serpent[21] and LLL[22]) that are designed to target the Ethereum Virtual Machine. Other platforms do not have programming languages that are specially designed for them. For instance, Chaincode (smart contracts in Hyperledger Fabric) can be written in Go, node.js and other programming languages. The topics related to development tools are only popular in Ethereum and Hyperledger Fabric because only these two platforms have representative development tools, i.e., Truffle for Ethereum and Hyperledger Composer for Hyperledger Fabric.

> The Application, API, Consensus and Network layer topics are discussed across the studied blockchain platforms, but exhibit different distributions in topic popularity.



(a) Bitcoin

(b) Ethereum

(c) Hyperledger Fabric

(d) Corda

Fig. 7. The absolute *impact* scores of the studied blockchain platforms, as compared with the trends of market prices of Bitcoin and Ethereum.

4.3.2 Topic Absolute Impact

We plotted the absolute impact of topics for the studied blockchain platforms in Fig. 7. As of November 2017, Bitcoin and Ethereum received absolute impact scores up to three orders of magnitude; The absolute impact scores of Hyperledger and Corda are two and one order(s) of magnitude, respectively.

The absolute impact of architectural layer topics exhibits an increasing trend, but with different speeds across blockchain platforms. The subplots in Fig. 7 suggest that the topics at each architectural layer across the studied blockchain platforms exhibit an increasing trend. We used the implementation of Cox Stuart trend test [26] in the `snpar` R package to determine if the increasing trend is statistically significant as shown in Table 4. In all topics except *Hyperledger Fabric: Application* and *Corda: Network*, the results reveal that the increasing trend in absolute impact is statistically significant. The studied blockchain platforms gained the absolute impact at different speeds.

The growth of absolute impact mirrors the growth of market prices of cryptocurrencies. As plotted in Fig. 7(a) and 7(b), the topic impact of Bitcoin and Ethereum grew with the market price of their corresponding cryptocurrency. The growth of topic impact and market price followed a similar pattern, but with different speeds. The coefficients of the Pearson's product-moment correlation suggest that the topic impact is strongly associated with the market price for Bitcoin (cor = 0.80, p-value < 2.2e-16) and Ethereum (cor = 0.90, p-value < 4.0e-12). The Bitcoin price grew around two times faster than the absolute impact of Bitcoin topics, while

[17] https://en.bitcoin.it/wiki/Bitcoin_Core_0.11_(ch_2):_Data_Storage

[18] https://ethereum.stackexchange.com/questions/28976/leveldb-in-geth-key-and-values

[19] https://hyperledger-fabric.readthedocs.io/en/release-1.4/ledger/ledger.html#blockchain

[20] https://docs.corda.net/node-database.html

[21] https://github.com/ethereum/serpent

[22] http://lll-docs.readthedocs.io/en/latest/lll_introduction.html

TABLE 4
The p-values of Cox Stuart trend tests with the hypothesis of increasing trends in the absolute impact of topics at each architectural layer ("**" indicates p-value < 0.01, "*" indicates p-value < 0.05).

	Bitcoin		Ethereum		Hyperledger Fabric		Corda	
Application	4.7e-03	**	3.1e-05	**	0.055		1.6e-02	*
API	1.5e-04	**	3.1e-05	**	9.8e-04	**	3.1e-02	*
VM PL			3.1e-05	**				
Consensus	3.5e-05	**	3.1e-05	**	9.8e-04	**	3.1e-02	*
Network	1.2e-02	*	3.1e-05	**	9.8e-04	**	0.063	
Dev Tools			3.1e-05	**	9.8e-04	**		

TABLE 5
The Person's product-moment coefficients of correlations between the absolute impact of topics at architectural layers and Bitcoin/Ethereum prices (p-value < 0.001).

	Bitcoin Price	Ethereum Price
Application	0.81	0.92
API	0.75	0.83
VM PL	-	0.87
Consensus	0.78	0.89
Network	0.77	0.88
Dev Tools	-	0.89

the growth rate of Ethereum price is approximately 13% of the growth rate of Ethereum topics' absolute impact.

The absolute impact of the Application layer topics has the strongest association with the market price of a cryptocurrency, while the topic impact at the API layer has the weakest association with the market price. Table 5 shows the Pearson's product-moment correlation coefficients between topic impact at each architectural layer and market price for Bitcoin and Ethereum. There is a general pattern of strong association (cor > 0.75, p-value < 0.001) between topic impact at each architectural layer and market price across Bitcoin and Ethereum. It is interesting that the topic impact at the Application layer is most strongly associated with the market price for Bitcoin and Ethereum, respectively, compared to other layers. Meanwhile, the topic impact at the API layer is most weakly associated with the market price for Bitcoin and Ethereum, respectively. It is likely that the price movements of cryptocurrencies attract more interest from users than developers, and users' interests might prompt more discussions around application-related topics.

The jumps in absolute impact coincided with the fluctuation of cryptocurrency price. The Bitcoin discussions on Stack Exchange experienced a "jump" in absolute topic impact in April 2013. Due to processing delays from payment processors, the Bitcoin price dropped from 266 USD to 76 USD before returning to 160 USD within six hours[23].

In addition to the market price, we observed other factors that may associate with the impact of discussions:

The absolute impact increased by order of magnitude following the introduction of a specialized Stack Exchange site on the domain. The Stack Exchange sites of Bitcoin and Ethereum had their first questions posted on August 30, 2011 and January 20, 2016, respectively. Consequently, Bitcoin reached its first sudden peak in September 2011, while Ethereum had a fast growth in January 2016.

Many jumps in the absolute impact coincide with news coverage. It is likely that if any news of a blockchain platform is widely distributed, the platform will attract great interest from the public, and the topic impact of that platform on Stack Exchange would jump. On December 5, 2013, The People's Bank of China prohibited Chinese financial institutions from using Bitcoin cryptocurrency. The absolute topic impact of Bitcoin experienced a "jump" in December of 2013 as shown in Fig. 7(a). In May 2016, a smart contract on the Ethereum blockchain named "The DAO" managed to gather 12.7 million Ethereum cryptocurrency, which is worth 150 million USD at the time, making it the biggest crowdfund ever. However, on June 17, 2016, an attacker successfully exploited a bug in the DAO project and put approximately 60 million USD under her control[24]. Finally, the Ethereum community decided to hard-fork the Ethereum blockchain and discarded the transaction involved in the attack. The absolute topic impact of Ethereum experienced a "jump" due to "The DAO" event.

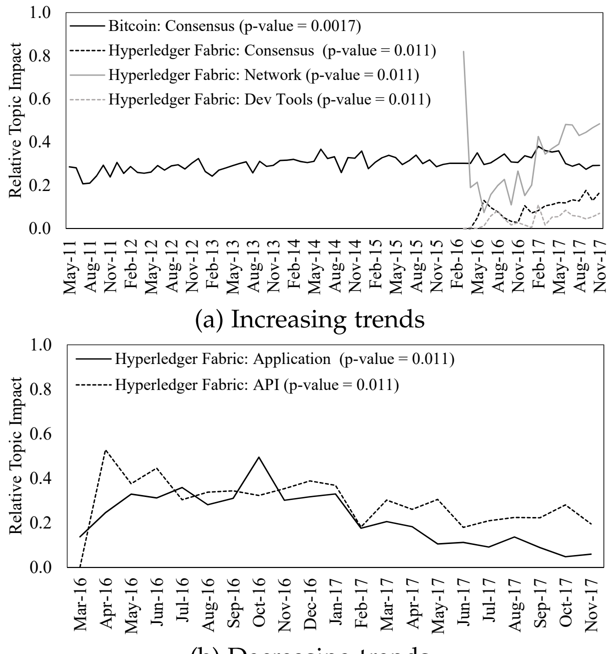
> The absolute impact of architectural layer topics exhibits an upward trend, but grows at different speeds across blockchain platforms.
>
> The growth of absolute impact mirrors the growth of cryptocurrency's market price, especially for the absolute impact of Application layer topics.
>
> Many jumps in absolute impact coincide with fluctuation of market price, the introduction of specific Stack Exchange site, and exposure of news.

4.3.3 *Topic Relative Impact*

We performed Cox Stuart trend test to assess whether there is an increasing or decreasing trend in the relative impact of topics at a particular architectural layer within each studied blockchain platform. Fig. 8 plots the relative impact of architectural-layer topics with statistically significant trends (p-value < 0.05). The results indicate that (1) although the absolute impact is increasing, the breakdown of absolute impact across architectural layers within each blockchain platform is relatively stable over time, especially for Ethereum and Corda; (2) for Bitcoin, the absolute impact of the Consensus layer topics grows faster than other layer topics over time; (3) for Hyperledger Fabric, the absolute impact of the Application and API layer topics grows slower, while the absolute impact of the Consensus,

[23] http://nymag.com/intelligencer/2013/04/inside-the-bitcoin-bubble-bitinstants-ceo.html?gtm=bottom&gtm=bottom

[24] http://www.coindesk.com/understanding-dao-hack-journalists



(a) Increasing trends

(b) Decreasing trends

Fig. 8. The relative *impact* scores of the architectural layer topics with statistically significant trends.

Network and Development Tools layers grows faster than the other layer topics over time. We further demystify below the statistically significant trends in relative impact.

The increasing trend in relative impact of *Consensus* layer topics for Bitcoin combines the decrease in relative impact of the *Mining* topic (p-value = 9.7e-09), and the increase in relative impact of the *Transaction* (p-value = 9.7e-09) and *Address* topics (p-value = 0.0032). The interest in the *Mining* topic grows slowly over time. Bitcoin miners gave way to large corporations. Cryptocurrency mining became more complicated after significant players came out. This might be associated with the decreasing trend in the relative impact of the *Mining* topic. On the other hand, the growth of interest in the *Transaction* topic appears to continue to increase over time. Bitcoin platform processes a growing number of transaction per day, but still amounts to approximately 0.7 transactions per second (TPS) [82]. The discussions of the *Transaction* could serve as inspiration for future research and practice.

The *Smart Contract* topics constitute the Application layer topics for Hyperledger Fabric. The impact of the *Smart Contract* topic grows faster than other topics initially since March 2016. We observed an inflection point of relative impact in October 2016. In fact, on September 16, 2016, Hyperledger Fabric released its v0.6.0-preview version. V0.6.0-preview release is the last release under the original architecture and implements the smart contract environment. All subsequent releases deliver on the v1.0 architecture[25]. The inflection point appears to lag behind the release of a new architecture. It is likely that the community will react to important release of the software, but the interest will fade away later.

Among the API layer topics in Hyperledger Fabric, the *Client* topic is most responsible for the decreasing trend on API layer topics. Within Hyperledger Fabric, the *Client* topic exhibited a downward trend starting with the greatest relative impact since its first appearance of the topic in April 2016. Note that Hyperledger Fabric launched its first release v0.1.0 on 16 February 2016[26]. Two months later, discussions of the *Client* topic appeared on Stack Overflow. At that time, the documentation for Hyperledger Fabric only included two sections related to its client (`Peer`) in the `README.md` file[27]. After a developer preview release (v0.5) was launched on 21 June 2016, the documentation for Hyperledger Fabric has continued to evolve, from a single `README.md` file to a structured website[28]. The decreasing trend might associate with the immaturity in documentation and lack of tutorials at the beginning. Generating clear and comprehensive documentation and tutorials may be useful to address this problem. The knowledge and experience sharing nature of Stack Exchange should motivate researchers to focus on innovative ways to generate and clarify documentation by utilizing the knowledge repositories.

> The breakdown of topic impact across architectural layer is stable over time except Hyperledger Fabric.
>
> For Ethereum discussions, the growth of interest slows down in the "Consensus: Mining" topic, but speed up in the "Consensus: Transaction" and "Consensus: Address" topics.
>
> For Hyperledger Fabric discussions, an inflection point appeared in the relative impact of the "Application: Smart Contract" topic, which might associate with its important release; The slow-down of interest in the "Client" topic might associate with the maturity of documentation.

5 Discussions

5.1 Implications

Community Evolution. Examining the results of the analysis, we note that the blockchain communities on Stack Exchange have been increasing significantly in volume. There appear no signs of stagnation in topic impact. Interestingly, the Ethereum community gains comparable topic impact to the Bitcoin community within two and half years, demonstrating the great potential of community growth.

The communities of the various blockchain platforms show different structures of interest in the discussed topics across architectural layers. The structure of interest is likely associated with the inherent features of each platform. For instance, the Ethereum community is more interested in topics at the Application layer compared to other layers; meanwhile, the *Smart Contract* topic, as an inherent feature of Ethereum, is most responsible for the popularity of Application layer topics. The structure of interest is also likely associated with the structure of community contributors. Future studies are needed to better characterize the contributors of the blockchain communities on Stack Exchange,

[25] https://fabricdocs.readthedocs.io/en/origin-v0.6/releases.html

[26] https://github.com/hyperledger-archives/fabric/releases/tag/v0.1.0

[27] https://github.com/hyperledger-archives/fabric/blob/edc391664ea81d987b1f58011fe9263cc3fc0fa3/README.md

[28] http://hyperledger-fabric.readthedocs.io/en/latest

e.g., the evolution of active contributors, participation patterns of contributors and their motivation of participation.

The Bitcoin and Ethereum communities are likely to be impacted by the fluctuation of cryptocurrency prices and external events. For instance, we observed that jumps in the topic impact of Bitcoin coincide with drops and rises in Bitcoin price; similarly, the topic impact of Ethereum went through a jump within "The DAO" event. Are the discussions of cryptocurrency communities on Stack Exchange merely hype-based, or offering technological contributions? Future in-depth studies are needed to determine if Stack Exchange discussions in other communities are impacted by such non-technical aspect as well or if our observations are specific to the blockchain communities.

Documentation Support. We notice that the *API* topic is commonly discussed across the studied blockchain platforms, and exhibits a high impact at the beginning since the topic first appeared. For Bitcoin and Ethereum, the impact of the *API* layer topics has the weakest association with cryptocurrency prices as compared to other layer topics. This phenomenon is likely associated with the immaturity in the documentation and lack of tutorials at the beginning, and lack of updates for documentation later on. Generating clear and comprehensive documentation and tutorials is vital to the continuous development of the blockchain communities. The knowledge and experience sharing nature of Stack Exchange should motivate researchers to focus on innovative ways to generate documentation by leveraging such rich knowledge repositories.

Transaction-level Tools. The *Transaction* topic is popular across the studied blockchain platforms in our dataset. Questions related to the *Transaction* topic include building, sending, analyzing and validating transactions, cryptographic primitives in transactions, transaction fees, etc. Failures in transactions negatively impact the reliability and usability of blockchain platforms. To improve the reliability and usability of blockchain platforms, it is essential to develop software engineering tools (e.g., transaction-level debuggers, logging, and testing tools) to help practitioners carefully trace transactions throughout their life cycle.

Security Concerns. We note that the *DAO* topic in Ethereum experienced a sharp decline in impact since October 2016. This decline happened three months after the DAO attack[29]. In June 2017, an attacker successfully exploited a bug in the DAO project and put approximately 60 million dollars under her control. Finally, the Ethereum community decided to hard-fork the Ethereum blockchain and discarded the transactions that are involved in the attack. Future research could be conducted on security analysis, vulnerability detection and security hardening for blockchain platforms. Testing techniques can also enhance the security and reliability of blockchain platforms. Blockchain specific testing techniques should be based on the decentralized and security-critical nature of blockchain platforms.

Smart Contract. Topics related to smart contracts are popular across the studied blockchain platforms in our dataset. The *Smart Contract* topics account in total for 18.4%, 28.4%, and 16.5% of the Ethereum, Corda and Hyperledger Fabric discussions. In contrast to traditional distributed applications, smart contracts are irreversible and immutable, i.e., smart contracts cannot be patched when bugs are detected. There is no way to patch a buggy smart contract without reversing the blockchain regardless of its popularity. Meanwhile, bugs in a smart contract might bring disastrous damages. Reasoning about the correctness of smart contracts before deployment is critical. High-level programming languages (e.g., Solidity[30] and Serpent[31]) have been designed to facilitate the development of smart contracts. Future research should take into consideration the techniques that are related to bug detection and localization for smart contract oriented programming languages.

5.2 Threats to Validity

Internal Validity. When performing our LDA computation, choosing the optimal number of topics (K) is difficult. To alleviate this threat, we used Röder et al.'s four-stage topic coherence pipeline to calculate the topic coherence for our built LDA models with different values of K. We selected the topic models with optimal K that produced topics with the highest coherence. In addition, LDA gives different results when running it several times on the same corpus since it is a probabilistic method. It is possible that our topic results are random to some degree. To mitigate this risk, we ran our LDA models three times, and compared the optimal number of topics and topics across each model. We found no significant differences in the optimal K between the LDA runs.

External Validity. Our potential threat to external validity is that our study only includes the dataset of Stack Overflow and two Stack Exchange sites. Although Stack Overflow and Stack Exchange are top development Q&A websites, further investigation could be conducted to include more sources.

6 Related Work

6.1 Blockchain Studies

Ecosystem. Previous studies investigated the factors that are associated with price movements of cryptocurrencies, including supply and demand, legal issues on adoption, macro-economic factors, and speculation and attractiveness of cryptocurrencies based on their potential [79]. These studies proposed models based on these factors to explore their relationships with the prices or trade volume of cryptocurrencies [16], [22], [39], [76]. Recent studies also investigated the relationship between cryptocurrency prices and social factors (e.g., social attractiveness) to predict the evolution of the cryptomarket and price fluctuations [31], [44], [63], [83]. Various metrics were leveraged to measure social factors, e.g., Google search term frequency [83] and posts on Twitter [63]. A few recent studies investigated the discussions in https://bitcointalk.org online forum to investigate the relationship between infrastructure evolution and community creation [4], self-organizing patterns in the community [4], sentiment analysis to predict price fluctuation [47], and sensemaking in the discussions [43].

[29] http://www.coindesk.com/understanding-dao-hack-journalists

[30] https://github.com/ethereum/solidity

[31] https://github.com/ethereum/serpent

Security and Privacy. Atzei et al. [8] analyzed the security vulnerabilities of Ethereum smart contracts, providing a taxonomy of vulnerabilities and attacks against Solidity, EVM, and blockchain. They defined 12 vulnerabilities among these components that pose risks to smart contract owners. Cook et al. utilized Atzei et al.'s vocabulary and provided a list of brief descriptions. They also proposed a monitoring and protection system to defend Solidity smart contracts. Gervais et al. [33] proposed a quantitative framework to analyze the security and performance implications of various consensus and network parameters of PoW blockchains. Based on the framework, they devised adversarial strategies for double-spending and selfish mining. Kosba et al. [50] addressed the lack of transactional privacy issue in blockchain platforms and proposed *Hawk*, a framework for building privacy-preserving smart contracts. Luu et al. [58] proposed *Elastico*, a secure sharding protocol for permissionless blockchains. The protocol increases the transaction throughput almost linear with the computational power of the blockchain network. For privacy-preserving payments, Heilman et al. [40] constructed off-chain payments with third party privacy; Green and Miers [34] proposed techniques to construct payment channels for decentralized payments.

Performance. Sompolinsky and Zohar [82] proposed the *GHOST* protocol, which changes the chain selection rule when a fork occurs and achieves a high transaction throughput. Kogias et al. [48] proposed *ByzCoin*, a Byzantine consensus protocol that leverages collective signing to achieve a transaction confirmation latency of under one minute. Bitcoin Lightning Network [78] and micro-payment channels [27] used off-chain transactions to improve the latency and throughput of the Bitcoin network. Eyal et al. [28] proposed *Bitcoin-NG*, a blockchain protocol based on the same trust model as Bitcoin. *Bitcoin-NG* achieves scalable latency and bandwidth by decoupling Bitcoin's blockchain operation.

Bugs. Luu et al. [57] investigated the security bugs of Ethereum smart contracts, proposed ways to enhance operation semantics of Ethereum, and built a symbolic execution tool called *Oyente* to find potential security bugs. Wan et al. [90] performed an empirical study on the bug characteristics of open source blockchain systems.

6.2 Studies of the Stack Exchange Community

An extensive literature used Stack Exchange to facilitate software engineering tasks. Bajaj et al. [10] studied common challenges and misconceptions amongst Web developers. Allamanis and Sutton [3] applied topic modeling on Stack Overflow questions and associated them with programming concepts and identifiers. Li et al. [53] performed an empirical study with 24 developers to determine the needs and challenges of developers when performing development tasks. Nasehi et al. [68] investigated what makes an effective code example through a qualitative analysis of Stack Overflow posts. Wang and Godfrey [91] analyzed iOS and Android developer questions on Stack Overflow to detect API usage obstacles. Azad et al. [9] created rules to predict API call usage by grouping API calls that are contained in positively voted answer posts on Stack Overflow. Treude and Robillard [89] presented an approach to automatically augment API documents with insight sentences from Stack Overflow. Mastrangelo et al. [61] studied issues encountered using Java API `sun.misc.Unsafe` by analyzing its usage patterns on Stack Overflow.

Other studies focus on investigating the characteristics of Stack Exchange discussions. Treude et al. [88] categorized the questions on Stack Overflow. Barua et al. [12] analyzed the textual contents and analyzed the topics and trends on Stack Overflow. Rosen and Shihab [81] studied mobile-related questions on Stack Overflow. Yang et al. [95] specifically studied security-related questions on Stack Overflow. Asaduzzaman et al. [7] analyzed unanswered questions on Stack Overflow and used a machine learning classifier to predict such questions.

6.3 LDA in Software Engineering

LDA has been previously employed to analyze software engineering data. Hindle et al. [41] applied LDA to the commit log messages in a version control system in an effort to determine what topics are being worked on by developers at any given time, and seeing how development trends are changing. Neuhaus and Zimmermann [69] applied LDA on a well-known vulnerability database to find the trends in specific security vulnerabilities over time. Thomas et al. [86] applied LDA to analyze software evolution, and later proposed a variant of LDA that can better detect topic trends in source code [87]. Garcia et al. [32] proposed a hierarchical clustering algorithm that utilizes LDA to extract concerns from identifiers and comments of the source code. Panichella et al. [77] proposed a solution called *LDA-GA*. They leveraged Genetic Algorithms (GA) to determine optimal configurations for LDA in software engineering tasks. Those tasks involve the analysis of source code, which is more repetitive and predictable as compared to natural language. Sophisticated information retrieval methods show rather low performance when applied on source code using parameters and configurations that were generally applicable for and tested on natural language corpora [77]. Barua et al. [12] used LDA to discover main discussion topics in Stack Overflow posts and analyzed the trends over time. Nguyen et al. proposed approaches based on LDA for bug localization [72] and duplicate bug report detection [73]. Rosen and Shihab [81] employed LDA-based topic models to summarize the mobile-related questions on Stack Overflow. Yang et al. [95] used LDA-GA to cluster security-related questions on Stack Overflow and investigated the popularity and difficulty of discovered topics.

7 Conclusion

This paper aims to discover blockchain-related topics on the Stack Exchange communities, the popular development Q&A websites with millions of active users. However, a simplistic use of LDA captures the topics in discussions blindly without keeping in mind the variety of the dataset and domain-specific concepts. We derived a reference architecture for the blockchain domain in an empirical data-driven fashion. We proposed an approach that combines balanced LDA (which ensures that the topics are balanced across the domain) with the reference architecture of a domain to capture. we conducted a quantitative and comparative analysis on blockchain-related posts across the architectural layers and across studied blockchain platforms where appropriate, characterized the breakdown and evolution of topics, and discussed their implications on future research efforts.

Acknowledgement

This research was partially supported by the National Key Research and Development Program of China (2018YFB1003904), NSFC Program (No. 61602403), Project of Science and Technology Research and Development Program of China Railway Corporation (P2018X002), and the Fundamental Research Funds for the Central Universities.

References

[1] A. Agrawal, W. Fu, and T. Menzies. What is wrong with topic modeling? and how to fix it using search-based software engineering. *Information and Software Technology*, 98:74–88, 2018.

[2] Alastria. Reference architecture of a blockchain. https://github.com/alastria/alastria-platform/blob/master/Requirements/Reference-architecture.md, 2018. Mar. 2019.

[3] M. Allamanis and C. Sutton. Why, when, and what: analyzing stack overflow questions by topic, type, and code. In *Working Conference on Mining Software Repositories*, pages 53–56. IEEE Press, 2013.

[4] J. V. Andersen and C. I. Bogusz. Patterns of self-organising in the bitcoin online community: Code forking as organising in digital infrastructure. In *International Conference on Information Systems*, 2017.

[5] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *13th EuroSys Conference*, page 30. ACM, 2018.

[6] S. Angelov, P. Grefen, and D. Greefhorst. A framework for analysis and design of software reference architectures. *Information and Software Technology*, 54(4):417–431, 2012.

[7] M. Asaduzzaman, A. S. Mashiyat, C. K. Roy, and K. A. Schneider. Answering questions about unanswered questions of stack overflow. In *Working Conference on Mining Software Repositories*, pages 97–100. IEEE Press, 2013.

[8] N. Atzei, M. Bartoletti, and T. Cimoli. A survey of attacks on ethereum smart contracts (sok). In *International Conference on Principles of Security and Trust*, pages 164–186. Springer, 2017.

[9] S. Azad, P. C. Rigby, and L. Guerrouj. Generating api call rules from version history and stack overflow posts. *ACM Transactions on Software Engineering and Methodology*, 25(4):29, 2017.

[10] K. Bajaj, K. Pattabiraman, and A. Mesbah. Mining questions asked by web developers. In *Working Conference on Mining Software Repositories*, pages 112–121. ACM, 2014.

[11] C. Ballard, C. Compert, T. Jesionowski, I. Milman, B. Plants, B. Rosen, H. Smith, et al. *Information governance principles and practices for a big data landscape*. IBM Redbooks, 2014.

[12] A. Barua, S. W. Thomas, and A. E. Hassan. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering*, 19(3):619–654, Jun 2014.

[13] S. Bird and E. Loper. Nltk: the natural language toolkit. In *Annual Meeting of the Association for Computational Linguistics*, page 31. Association for Computational Linguistics, 2004.

[14] D. M. Blei and J. Lafferty. Topic models. text mining: theory and applications, 2009.

[15] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[16] J. Bouoiyour and R. Selmi. What does bitcoin look like? *Annals of Economics & Finance*, 16(2), 2015.

[17] L. Bovens and S. Hartmann. *Bayesian epistemology*. Oxford University Press on Demand, 2003.

[18] S. Brakeville and B. Perepa. Blockchain basics: Introduction to distributed ledgers. https://developer.ibm.com/tutorials/cl-blockchain-basics-intro-bluemix-trs/, 2018. Mar. 2019.

[19] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn. Corda: An introduction. *R3 CEV, August*, 2016.

[20] C. Cachin. Architecture of the hyperledger blockchain fabric. *IBM Research*, 2016.

[21] D. Chappelle. Big Data and Analytics Reference Architecture. https://www.oracle.com/technetwork/topics/entarch/oracle-wp-big-data-refarch-2019930.pdf, 2013. Mar. 2019.

[22] J. Chiu and T. V. Koeppl. The economics of cryptocurrencies–bitcoin and beyond. *Available at SSRN 3048124*, 2017.

[23] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole, and M. Bone. The concept of reference architectures. *Systems Engineering*, 13(1):14–27, 2010.

[24] CoinMarketCap. Crypto-Currency Market Capitalizations. https://coinmarketcap.com, 2018. Mar. 2019.

[25] C. community. Colored Coins. http://coloredcoins.org, 2013. Mar. 2019.

[26] D. R. Cox and A. Stuart. Some quick sign tests for trend in location and dispersion. *Biometrika*, 42(1/2):80–95, 1955.

[27] C. Decker and R. Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*, pages 3–18. Springer, 2015.

[28] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *NSDI*, pages 45–59, 2016.

[29] B. Gallagher. Using the architecture tradeoff analysis method to evaluate a reference architecture: A case study. Technical Report CMU/SEI-2000-TN-007, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2000.

[30] M. Galster, S. Angelov, S. Martínez-Fernández, and D. Tofan. Reference architectures and scrum: friends or foes? In *11th Joint Meeting on Foundations of Software Engineering*, pages 896–901. ACM, 2017.

[31] D. Garcia and F. Schweitzer. Social signals and algorithmic trading of bitcoin. *Royal Society open science*, 2(9):150288, 2015.

[32] J. Garcia, D. Popescu, C. Mattmann, N. Medvidovic, and Y. Cai. Enhancing architectural recovery using concerns. In *26th IEEE/ACM International Conference on Automated Software Engineering*, pages 552–555. IEEE, 2011.

[33] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun. On the security and performance of proof of work blockchains. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 3–16. ACM, 2016.

[34] M. Green and I. Miers. Bolt: Anonymous payment channels for decentralized currencies. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 473–489. ACM, 2017.

[35] T. L. Griffiths, M. Steyvers, and J. B. Tenenbaum. Topics in semantic representation. *Psychological review*, 114(2):211, 2007.

[36] A. Grosskurth and M. W. Godfrey. A reference architecture for web browsers. In *21st IEEE International Conference on Software Maintenance*, pages 661–664. IEEE Computer Society, 2005.

[37] T. O. Group. SOA Reference Architecture. http://www.opengroup.org/soa/source-book/soa_refarch/index.htm, 2011. Mar. 2019.

[38] A. E. Hassan and R. C. Holt. A reference architecture for web servers. In *7th Working Conference on Reverse Engineering*, pages 150–159. IEEE, 2000.

[39] A. S. Hayes. Cryptocurrency value formation: An empirical study leading to a cost of production model for valuing bitcoin. *Telematics and Informatics*, 34(7):1308–1321, 2017.

[40] E. Heilman, F. Baldimtsi, and S. Goldberg. Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. In *International Conference on Financial Cryptography and Data Security*, pages 43–60. Springer, 2016.

[41] A. Hindle, M. W. Godfrey, and R. C. Holt. What's hot and what's not: Windowed developer topic analysis. In *IEEE International Conference on Software Maintenance*, pages 339–348. IEEE, 2009.

[42] M. Hoffman, F. R. Bach, and D. M. Blei. Online learning for latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 856–864, 2010.

[43] E. Jahani, P. M. Krafft, Y. Suhara, E. Moro, and A. S. Pentland. Scamcoins, s*** posters, and the search for the next bitcoin$^{\text{TM}}$: Collective sensemaking in cryptocurrency discussions. In *Proceedings of the ACM on Human-Computer Interaction - CSCW*, New York, NY, USA, 2018. ACM.

[44] J. Kaminski. Nowcasting the bitcoin market with twitter signals. *arXiv preprint arXiv:1406.7577*, 2014.

[45] R. Kazman, M. Klein, and P. Clements. Atam: Method for architecture evaluation. Technical Report CMU/SEI-2000-TR-004, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2000.

[46] M. B. Kelly, J. S. Alexander, B. Adams, and A. E. Hassan. Recovering a balanced overview of topics in a software domain. In *11th IEEE Working Conference on Source Code Analysis and Manipulation*, pages 135–144. IEEE, 2011.

[47] Y. B. Kim, J. G. Kim, W. Kim, J. H. Im, T. H. Kim, S. J. Kang, and C. H. Kim. Predicting fluctuations in cryptocurrency transactions based on user comments and replies. *PloS one*, 11(8):e0161197, 2016.

[48] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th USENIX Security Symposium*, pages 279–296, 2016.

[49] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *IEEE Symposium on Security and Privacy*, pages 839–858, May 2016.

[50] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *IEEE Symposium on Security and Privacy*, pages 839–858. IEEE, 2016.

[51] A. Kuhn, S. Ducasse, and T. Gírba. Semantic clustering: Identifying topics in source code. *Information and Software Technology*, 49(3):230–243, 2007.

[52] D. Kum, G.-M. Park, S. Lee, and W. Jung. Autosar migration from existing automotive software. In *International Conference on Control, Automation and Systems*, pages 558–562. IEEE, 2008.

[53] H. Li, Z. Xing, X. Peng, and W. Zhao. What help do developers seek, when and how? In *Working Conference on Reverse Engineering*, pages 142–151. IEEE, 2013.

[54] M. Linares-Vásquez, B. Dit, and D. Poshyvanyk. An exploratory analysis of mobile development issues using stack overflow. In *Working Conference on Mining Software Repositories*, pages 93–96. IEEE Press, 2013.

[55] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf. Nist cloud computing reference architecture. *NIST special publication*, 500(2011):1–28, 2011.

[56] S. K. Lukins, N. A. Kraft, and L. H. Etzkorn. Source code retrieval for bug localization using latent dirichlet allocation. In *Working Conference on Reverse Engineering*, pages 155–164. IEEE, 2008.

[57] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor. Making smart contracts smarter. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 254–269. ACM, 2016.

[58] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena. A secure sharding protocol for open blockchains. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 17–30. ACM, 2016.

[59] S. Martínez-Fernández, C. P. Ayala, X. Franch, and H. M. Marques. Benefits and drawbacks of reference architectures. In *European Conference on Software Architecture*, pages 307–310. Springer, 2013.

[60] S. Martínez-Fernández, C. P. Ayala, X. Franch, and E. Y. Nakagawa. A survey on the benefits and drawbacks of autosar. In *1st International Workshop on Automotive Software Architecture*, pages 19–26. ACM, 2015.

[61] L. Mastrangelo, L. Ponzanelli, A. Mocci, M. Lanza, M. Hauswirth, and N. Nystrom. Use at your own risk: The java unsafe api in the wild. In *Proceedings of OOPSLA 2015 (International Conference on Object Oriented Programming Systems Languages and Applications)*, pages 695–710. ACM Press, 2015.

[62] G. Mathew, A. Agrawal, and T. Menzies. Finding trends in software research. *IEEE Transactions on Software Engineering*, pages 1–1, 2018.

[63] M. Matta, I. Lunesu, and M. Marchesi. Bitcoin spread prediction using social and web search media. In *UMAP Workshops*, pages 1–10, 2015.

[64] O. Maxim. Towards common blockchain architecture - an "ISO OSI for blockchain" primer. https://medium.com/pandoraboxchain/towards-common-blockchain-architecture-d8a5f0e2541e, 2017. Mar. 2019.

[65] J. Meier, A. Homer, D. Hill, J. Taylor, P. Bansode, L. Wall, R. Boucher, and A. Bogawat. *Microsoft application architecture guide: patterns and practices*. Microsoft Press, Redmond, 2009.

[66] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[67] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Bitcoin White Paper*, 2008.

[68] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns. What makes a good code example?: A study of programming q&a in stackoverflow. In *IEEE International Conference on Software Maintenance*, pages 25–34. IEEE, 2012.

[69] S. Neuhaus and T. Zimmermann. Security trend analysis with cve topic models. In *IEEE 21st International Symposium on Software Reliability Engineering*, pages 111–120. IEEE, 2010.

[70] D. Newman, C. Chemudugunta, P. Smyth, and M. Steyvers. Analyzing entities and topics in news articles using statistical topic models. In *International Conference on Intelligence and Security Informatics*, pages 93–104. Springer, 2006.

[71] D. Newman, J. H. Lau, K. Grieser, and T. Baldwin. Automatic evaluation of topic coherence. In *Human Language Technologies: Annual Conference of the North American Chapter of ACL*, pages 100–108. Association for Computational Linguistics, 2010.

[72] A. T. Nguyen, T. T. Nguyen, J. Al-Kofahi, H. V. Nguyen, and T. N. Nguyen. A topic-based approach for narrowing the search space of buggy files from a bug report. In *26th IEEE/ACM International Conference on Automated Software Engineering*, pages 263–272. IEEE Computer Society, 2011.

[73] A. T. Nguyen, T. T. Nguyen, T. N. Nguyen, D. Lo, and C. Sun. Duplicate bug report detection with a combination of information retrieval and topic modeling. In *27th IEEE/ACM International Conference on Automated Software Engineering*, pages 70–79. ACM, 2012.

[74] OATH. OATH Reference Architecture. https://openauthentication.org/wp-content/uploads/2015/09/ReferenceArchitectureVersion2.pdf, 2013. Mar. 2019.

[75] L. B. R. Oliveira and E. Y. Nakagawa. A service-oriented reference architecture for software testing tools. In *European Conference on Software Architecture*, pages 405–421. Springer, 2011.

[76] B. Ong, T. M. Lee, G. Li, and D. L. K. Chuen. Evaluating the potential of alternative cryptocurrencies. In *Handbook of digital currency*, pages 81–135. Elsevier, 2015.

[77] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshynanyk, and A. De Lucia. How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In *35th International Conference on Software Engineering*, pages 522–531. IEEE, 2013.

[78] J. Poon and T. Dryja. The bitcoin lightning network: Scalable off-chain instant payments. *Bitcoin Lightning Network White Paper*, 2016.

[79] O. Poyser. Exploring the determinants of bitcoin's price: an application of bayesian structural time series. *arXiv preprint arXiv:1706.01437*, 2017.

[80] M. Röder, A. Both, and A. Hinneburg. Exploring the space of topic coherence measures. In *International Conference on Web Search and Data Mining*, pages 399–408. ACM, 2015.

[81] C. Rosen and E. Shihab. What are mobile developers asking about? a large scale study using stack overflow. *Empirical Software Engineering*, 21(3):1192–1223, Jun 2016.

[82] Y. Sompolinsky and A. Zohar. Accelerating bitcoin's transaction processing. *Cryptology ePrint Archive*, 2013.

[83] Y. Sovbetov. Factors influencing cryptocurrency prices: Evidence from bitcoin, ethereum, dash, litcoin, and monero. *Journal of Economics and Financial Analysis*, 2(2):1–27, 2018.

[84] C. Staff. The Ashley Madison hack...in 2 minutes. http://money.cnn.com/2015/08/24/technology/ashley-madison-hack-in-2-minutes, 2015. Mar. 2019.

[85] M. Swan. *Blockchain: Blueprint for a new economy*. O'Reilly Media, Inc., 2015.

[86] S. W. Thomas, B. Adams, A. E. Hassan, and D. Blostein. Validating the use of topic models for software evolution. In *10th IEEE Working Conference on Source Code Analysis and Manipulation*, pages 55–64. IEEE, 2010.

[87] S. W. Thomas, B. Adams, A. E. Hassan, and D. Blostein. Modeling the evolution of topics in source code histories. In *Working Conference on Mining Software Repositories*, pages 173–182. ACM, 2011.

[88] C. Treude, O. Barzilay, and M.-A. Storey. How do programmers ask and answer questions on the web?: Nier track. In *33rd International Conference on Software Engineering*, pages 804–807. IEEE, 2011.

[89] C. Treude and M. P. Robillard. Augmenting api documentation with insights from stack overflow. In *38th International Conference on Software Engineering*, pages 392–403. IEEE, 2016.

[90] Z. Wan, D. Lo, X. Xia, and L. Cai. Bug characteristics in blockchain systems: a large-scale empirical study. In *International Conference on Mining Software Repositories*, pages 413–424. IEEE, 2017.

[91] W. Wang and M. W. Godfrey. Detecting api usage obstacles: A study of ios and android developer questions. In *Working Conference on Mining Software Repositories*, pages 61–64. IEEE Press, 2013.

[92] M. Weyrich and C. Ebert. Reference architectures for the internet of things. *IEEE Software*, 33(1):112–116, 2016.

[93] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151:1–32, 2014.

[94] D. Xiao. The Four Layers of the Blockchain. https://medium.com/@coriacetic/the-four-layers-of-the-blockchain-dc1376efa10f, 2016. Mar. 2019.

[95] X. Yang, D. Lo, X. Xia, Z. Wan, and J. Sun. What security questions do developers ask? a large-scale study of stack overflow posts. *Journal of Computer Science and Technology*, 31(5):910–924, 2016.

Appendix A
Data Collection

Collecting Stack Overflow Dataset Fig. 9 depicts our process of data collection. The process is composed of the following steps:

Step 1: Searching for tags using an initial set of blockchain keywords. We composed an initial set of blockchain keywords by considering blockchain platforms, protocols, infrastructures, and development tools. Table 6 shows the initial set of the used blockchain-related keywords. While the set of keywords in Table 6 might not be exhaustive, we believe that the set covers the most popular blockchain-powered cryptocurrencies and blockchain platforms. The list of blockchain-powered cryptocurrencies was derived from the most popular cryptocurrencies with large market capitalization. The list of blockchain platforms was derived from various online sources. We cross referenced the keywords to see their relevance on Stack Overflow.

TABLE 6
Initial keywords that are used to identify blockchain posts.

General	Cryptocurrency	Platform
blockchain	bitcoin	bigchaindb
	dogecoin	corda
	ethereum	eris
	iota	hyperledger
	litecoin	monax
		quorum

We began by searching through all the Stack Overflow posts that contain any of the initial set of keywords shown in Table 6 as tags. Tags are keywords that users attribute with their posts. For example, a post about Bitcoin address form validation issue has the tags "bitcoin", "javascript", "php" and "validation" attributed to it. We then extracted all the tags associated with each of those posts. We ended up with an expanded set of keywords from those posts.

Step 2: Removing irrelevant tags. Our goal is to use the expanded set of keywords to identify a larger set of blockchain posts. However, this approach may introduce noise, i.e., posts with tags that do not necessarily relate to blockchain. Given that an example post has the tags "bitcoin", "javascript", "php" and "validation", we would further add posts with any of these 4 tags. However, "javascript" posts are likely to refer to other development issues related to JavaScript. In such a case, such posts would introduce considerable noise into our dataset.

In order to determine exclusive tags related to blockchain, we use a Tag Exclusivity Threshold (TET) value, which is computed as follows:

$$TET_{tag} = \frac{\textit{Number of blockchain posts}}{\textit{Number of total posts}} \tag{10}$$

For each tag, the *Number of blockchain posts* denotes the number of posts with that tag which contain at least one of the initial set of blockchain keywords from Table 6; the *Number of total posts* represents the total number of posts with that tag. A similar approach is proposed [81] to determine exclusively mobile-related tags. We ranked tags according to their TET_{tag} values. We then manually verified the tags from top to bottom and picked the tags that are related exclusively to blockchain.

Once we filtered out the irrelevant tags using the TET value, we ended up with some tags with very few posts associated with them. Those tags were related to very specific problems. For example, some specific tags (e.g., populus) were only used in one post which happen to be blockchain-related. In such case, the TET value of those tags is equal to 1. However, incorporating such tags may not be very helpful. Thus, we use another threshold, Tag Significance Threshold (TST), to determine the significance of each tag. The TST value is measured as:

$$TST_{tag} = \frac{\textit{Number of blockchain posts}}{\textit{Number of blockchain posts with the most popular tag}} \tag{11}$$

For each tag, the *Number of blockchain posts* denotes the total number of posts with that tag which contain at least one of the initial set of blockchain keywords from Table 6; the *Number of blockchain posts with the most popular tag* is the blockchain posts with the tag "bitcoin", containing 1,094 posts. We ranked the tags according to their TST_{tag} values. We then manually verified the tags from top to bottom and picked the tags that are related significantly to blockchain.

Table 7 shows the final list of 28 filtered tags and their TET and TST values respectively.

TABLE 7
List of tags that are used to identify blockchain-related posts.

Tags	TET	TST
bitcoin	100.0%	100.0%
blockchain	100.0%	84.6%
hyperledger	40.3%	73.1%
ethereum	100.0%	64.6%
solidity	76.7%	22.4%
corda	100.0%	17.4%
bitcoind	76.9%	10.7%
smartcontracts	87.4%	10.1%
truffle	73.3%	8.2%
web3js	80.3%	6.5%
web3	75.5%	4.5%
bitcoinj	62.2%	4.1%
go-ethereum	92.7%	3.7%
coinbase-php	39.4%	3.0%
blockchain.info-api	93.8%	2.9%
bitcore	64.0%	2.3%
iota	100.0%	1.9%
dogecoin	100.0%	1.7%
bitcoin-testnet	84.2%	1.7%
consensys-truffle	64.3%	1.3%
bigchaindb	100.0%	1.3%
eris	100.0%	0.9%
metamask	88.9%	0.8%
geth	87.5%	0.7%
nbitcoin	100.0%	0.6%
evm	85.7%	0.6%
ether	57.1%	0.6%
embark	71.4%	0.6%

Step 3: Filtering posts with filtered tags. Once all of the blockchain-related tags were identified, we extracted all the posts of the questions that contained one of these blockchain tags. The question posts make up our Stack Overflow dataset in our experiments. In total, our Stack Overflow dataset consists of 3,830 question posts.

We then extracted tags of all the blokchain posts on Stack Overflow and found a total of 1,103 tags. We then examined the top ranked tags and found that the most

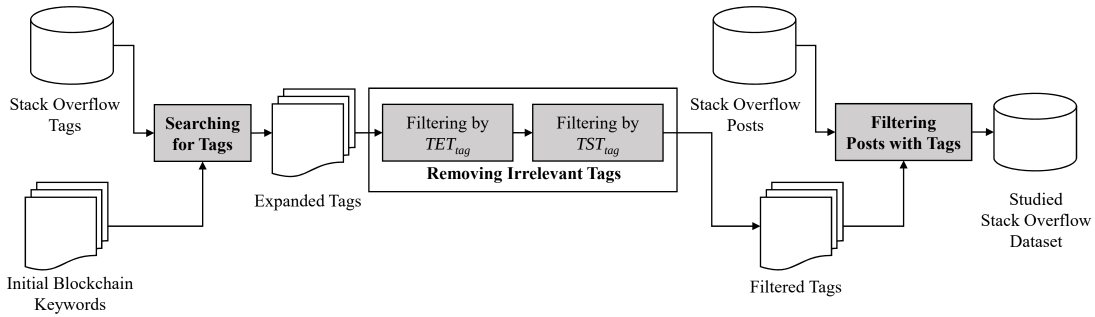


Fig. 9. Process of collecting Stack Overflow dataset.

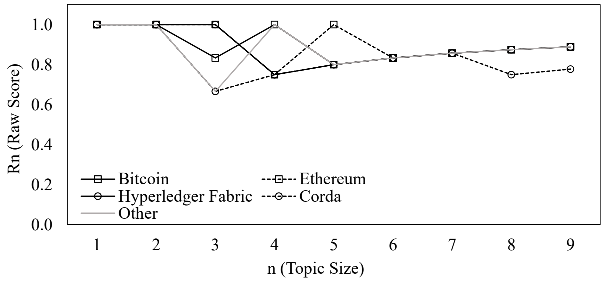


Fig. 10. Raw score R_n vs. topic size n, larger numbers indicate higher stability of generated topics.

popular blockchain topics on Stack Overflow are related to four blockchain platforms: Bitcoin, Ethereum, Hyperledger Fabric and Corda.

When extracting dataset, we leveraged tags to determine the set of blockchain-related posts. In some cases, our tags may not be able to capture a blockchain-related post. To alleviate this threat, we expanded our initial tags with those associated tags that coexist in the posts with the initial tags; we further filtered expanded tags by using thresholds to weed out noise.

Collecting Stack Exchange Dataset In order to include more blockchain-related posts in our experiments, we tried to identify specific blockchain-related communities on Stack Exchange.

We searched through all the Stack Exchange communities that focus on the most popular blockchain platforms on Stack Overflow. We ended up with two Stack Exchange communities related to blockchain, namely the Bitcoin and Ethereum communities. We extracted question posts from the data dump of the two communities (`Posts.xml`). The question posts of the two communities comprise our Stack Exchange dataset. In total, our Stack Exchange corpus contains 16,934 question posts from the Bitcoin Stack Exchange community and 11,971 question posts from the Ethereum Stack Exchange community.

Post Extraction We separated each individual post into a record in our database. We ended up with a total of 32,735 records. In each record, we included the textual content of each post including the title and body; we also included the post metadata including the identifier, the timestamp, the user-specified tags of that post.

Preprocessing We performed our data preprocessing in five steps to filter out irrelevant information in the textual content for each post before building the topic models.

Step 1: Discard code snippets. Code snippets in the posts are enclosed in `<code>` HTML tags. Code snippets usually contain source code syntax and keywords. They do not help topic models to identify useful topics and might introduce noise into our analysis [12].

Step 2: Remove HTML tags and URLs. We removed all HTML tags (e.g., `<p>`, `<li>` and `<a>`) and URLs since they do not have useful information for the topic models.

Step 3: Remove number, punctuation marks and other non-alphabetic characters.

Step 4: Lemmatize. We applied the NLTK WordNet Lemmatizer [66], which maps words to their base form (e.g., "makes" and "transactions" get mapped to "make" and "transaction").

Step 5: Remove stopwords. Finally, we removed common English-language stopwords by using the NLTK stopwords corpus [13].

We attempted to minimize the internal threats to validity by using mature tools for extracting and preprocessing our dataset (i.e., Python XML parser `lxml.etree`, Python NLTK Wordnet Lemmatizer `nltk.wordnet.WordNetLemmatizer` and NLTK Corpus Stopwords `nltk.corpus.stopwords`), running LDA (i.e., *gensim* Python library), and computing our metrics of interest (i.e., the R programming environment).

Appendix B
Stability of LDA Models

We conducted a case study to evaluate the stability of our LDA models due to the ordering of our training data. We leveraged the metric, raw score R_n, as defined in Agrawal et al's work [1]. R_n denotes the *median of Jaccard Similarity scores of n-word topics* from the two closest topics of multiple LDA runs; n is size of each topic - number of keywords that are used to represent each topic. For each corpus, we executed gensim LDA with optimal K to build topic models multiple times; before execution, we shuffled the training data. Fig. 10 plots n vs. R_n of generated LDA models. We observed that the average R_n for each corpus is above 82%; at worst case, the R_n is 67%.

Appendix C
Topics Discovered by LDA and Their Trends

Table 8 shows the discovered LDA topics from the studied corpora. Table 9 shows the trendlines of each topic in

TABLE 8
Topics discovered by LDA in Bitcoin, Ethereum, Hyperledger Fabric, Corda and non-platform specific corpora.

Topic Name	Percent	Top LDA Keywords
Bitcoin ($K = 10$)		
Crytpocurrency	16.6%	bitcoin bitcoins account btc exchange wallet address price money coin currency way user payment know
Network	15.2%	bitcoin node network data question value time full cpp limit alice system attack peer
Client	12.8%	bitcoin error get file core bitcoind run version command running problem code following test daemon
Address	12.1%	address key private bitcoin public code script get generate string output input signature create byte
Transaction	11.8%	transaction fee address bitcoin balance input send btc get sent unconfirmed help output wallet amount
Mining	10.0%	mining miner pool mine know difficulty bitcoin coin power get share process altcoin hash question
Wallet (Electrum)	6.3%	wallet electrum create seed bitcoin password new find code passphrase word trezor bitcoinj way multibit
Storage	6.1%	block chain transaction network hash miner new node fork nonce work proof valid find genesis
Web API	5.1%	api server data bitcoin web code user node php application com make request client work
Wallet (blockchain.info)	4.1%	blockchain coinbase info time day order btc wallet take long last coin way data back
Ethereum ($K = 10$)		
Smart Contract	18.4%	contract function smart code call solidity address deploy test deployed truffle work get token method
Client	15.5%	geth node block mining ethereum running start parity run blockchain peer sync command network get
Cryptocurrency	13.9%	ethereum token transaction question way possible blockchain coin key know make user
Wallet	10.7%	wallet ether eth account token address ethereum balance send transfer mist get sent help see
Account	9.8%	account private ethereum network file geth key node metamask create blockchain test created connect json
Solidity	8.1%	function value event return type string get solidity array name data code input address call
Storage	7.7%	data user blockchain contract store way ethereum smart dapp storage access stored mapping variable know
Transaction	7.4%	transaction gas block time number value get limit hash mined error price state send nonce
Truffle	6.5%	error truffle node testrpc file run following npm module get ethereum command install running version
DAO	2.0%	ethereum chain proof attack blockchain protocol fork hard dao difference scrypt network algorithm stake consensus
Hyperledger Fabric ($K = 15$)		
Peer (Runtime)	18.1%	peer hyperledger fabric chaincode network block error transaction docker blockchain make get multiple following application
Network	17.9%	fabric hyperledger composer docker network error following file app peer sample node get application transaction
Chaincode (Query/Deploy/Invoke)	13.6%	chaincode hyperledger error fabric blockchain query deploy data code get invoke following transaction container docker
Identity	11.4%	error composer network identity fabric hyperledger failed following command chaincode business participant get deploy issue
Peer (Command Line)	8.9%	fabric hyperledger peer error command file docker node network user attribute example code log following
Transaction	5.8%	transaction client hyperledger blockchain asset example composer fabric participant com sarama function broker metadata name
Channel (Setup)	5.6%	hyperledger fabric channel composer peer rest setup network sdk server org api docker new add
Orderer	5.3%	peer channel orderer error fabric network cli node chaincode config hyperledger utc name command client
Composer (Playground)	3.4%	error composer node version hyperledger fabric file playground module get blockchain rest following server tutorial
Storage	2.4%	example container com fabric network orderer hyperledger error couchdb host docker peer first get cli
Channel (Runtime)	2.3%	channel hyperledger contract user data ledger fabric smart blockchain network chaincode quantity function distributor peer
Chaincode (Programming)	1.9%	error chaincode debu gopath src file opt command utc test install docker composer code chain
Composer (REST Server)	1.2%	node module composer lib rest server version home nvm router express brankoterzic index error hyperledger
Fabric CA	1.1%	example com cert pem server user admin npm key crt install node tlsca fabric cacerts
Java SDK	1.1%	java org gradle internal service server execute daemon launcher file fabric get sdk exec defaultlookupcontext
Corda ($K = 5$)		
Transaction	30.0%	node transaction corda state flow contract party example notary way bank network multiple cash output
CorDapp	28.4%	corda error node kotlin code file run gradle tutorial issue flow cordapp following build template
State	21.1%	corda state node service code oracle data com class flow cordapp kryo core cash error
Storage	13.2%	corda node jar transaction java rxjava data party custom net table onnext cordapersistence internal database
Client	7.4%	kryo java jar corda com esotericsoftware net core node serializers client read objectfield rpc org
Non-Platform Specific ($K = 5$)		
PHP Library for Coinbase API	30.9%	api coinbase error code get blockchain file php following account request call work wallet response
Coinbase Transaction	21.4%	blockchain data block user new create transaction customer coinbase work application make technology
Node.js Library for Coinbase API	19.0%	error blockchain get node command coinbase service following new api create bluemix know work
Python Library for Coinbase API	18.0%	api coinbase transaction price get wallet code blockchain python account return way currency client work
Hash Functions	10.7%	blockchain block get hash chain key value way work function data server find send transaction

the third column, using the relative *impact* metric (7). To further analyze the topic trends in more detail, we used the implementation of Cox Stuart trend test [26] in the `snpar` R package to determine the trend of each topic's relative impact. We checked whether each topic's impact is *increasing* or *decreasing* over time to a statistically significant degree at a 95% confidence level. The results of Cox Stuart trend test are shown in the second column of Table 9. We find that 16 topics have an increasing trend, 6 topics have a decreasing trend, and 23 topics have a constant trend (i.e., neither increasing or decreasing to a statistically significant degree).

Appendix D
Representative Example Posts for Discovered Topics

Application layer. Out of the discovered 45 topics, 10 topics are specifically at the application layer of our derived reference architecture, namely: ① *Cryptocurrency*, ② *Wallet (Electrum)*, ③ *Wallet (blockchain.info)* for Bitcoin; ④ *Smart Contract*, ⑤ *Cryptocurrency*, ⑥ *Wallet*, ⑦ *DAO* for Ethereum; ⑧ *Chaincode (Query/Deploy/Invoke)*, ⑨ *Chaincode (Programming)* for Hyperledger Fabric; ⑩ *CorDapp* for Corda. Some example posts from these topics are:

> *electrum wallet jsonRPC authentication*
> i am trying to use electrum rpc, it is giving me authentication error. i have tried user pass via Basic Authentication on Linux bash and via php but non of them works.
>
> Dominant Topic: *Bitcoin/Wallet (Electrum)*

> *Upgrading smart contract in ethereum*
> I am trying to write upgradable smart contract in ethereum. Can anyone give an example of upgradable smart contract in ethereum and accessing data.
>
> Dominant Topic: *Ethereum/Smart Contract*

> *Hyperledger Fabric chaincode Invoke function return values*
> I understand that Chaincode Invoke function is asynchronous and cannot convey success/failure of ledger modification unless consensus is completed. However what about simple validation errors caught before invocation of any chaincode stub APIs? There should be a way to return error to caller in case of validation failure. Otherwise what is use of return value of function.
>
> Dominant Topic: *Hyperledger Fabric/Chaincode (Query/Deploy/ Invoke)*

> *Unique Identifier in Obligation Cordapp*
> I have replicated the Obligation Cordapp Transfer functionality and i am struck with the linearId of Unique Identifier. I have successfully exercised the Issue Cordapp and for transfer of Obligation, i have provided the flow command with linearId of generated Obligation. The parameter which i am passing through linearId is interpreted as the External id [argument in UniqueIdentifier] instead of id and so it is unable to find the Obligation to transfer.
>
> Dominant Topic: *Corda/CorDapp*

API layer. We discover 10 API layer topics in our dataset, including ① *Client*, ② *Web API* for Bitcoin, ③ *Client* for Ethereum, ④ *Peer (Runtime)*, ⑤ *Peer (Command Line)*, ⑥ *Java SDK* for Hyperledger Fabric, ⑦ *Client* for Corda, ⑧ *PHP library*, ⑨ *Node.js library*, ⑩ *Python library* for non-platform specific. The following example post is from these topics:

> *Bitcoind daemon - network synchronizing error*
> I have bitcoind installed on a VPS debian server, i started bitcoind 2 days ago for synchronizing with bitcoin network but i got the following error today. I have no idea what this error mean, or how can i resolve this kind of error.
>
> Dominant Topic: *Bitcoin/Client*

VM programming language layer. We find 1 VM programming language layer topic in our dataset, ① *Solidity* for Ethereum. The following example post is from these topics:

> *Division in Ethereum Solidity*
> I am creating a contract that issues tokens. I would like a account that holds tokens to be able to check what percentage they own out of all the tokens issued. I know that Ethereum has not implemented floating point numbers yet. What should I do?
>
> Dominant Topic: *Ethereum/Solidity*

Consensus layer. We find 11 consensus layer topics in our dataset, namely: ① *Address*, ② *Transaction*, and ③ *Mining* for Bitcoin, ④ *Account*, ⑤ *Transaction* for Ethereum, ⑥ *Transaction*, ⑦ *Orderer* for Hyperledger Fabric, ⑧ *Transaction*, ⑨ *State* for Corda, ⑩ *Transaction* for Coinbase API, and ⑪ *Hash Functions* for non-platform specific. Example post is:

> *launching hyperledger orderer fails*
> The official hyperledger fabric v1.0.0 gives a simple demo by using docker. Here is the link.
> What i am doing is to avoid docker and directly run them on the machine. However, when i try to launch the orderer using the following cmd and environment variables, the bash console reports the failure of loading config from orderer.
>
> Dominant Topic: *Hyperledger Fabric/Orderer*

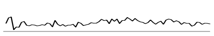
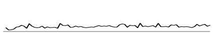
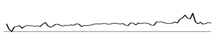
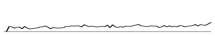
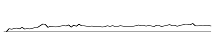
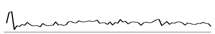
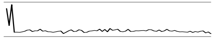
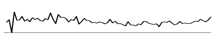
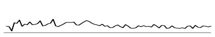
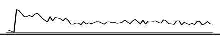
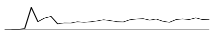
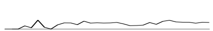
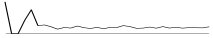
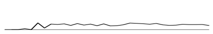
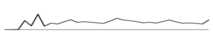
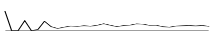
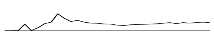
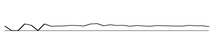
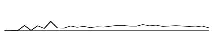
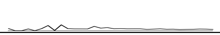
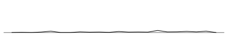
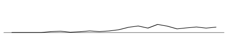
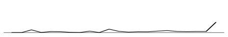
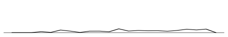
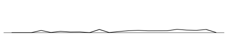
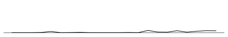
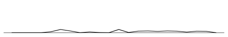
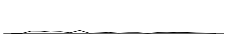
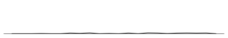
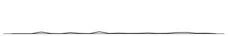
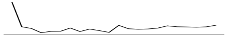
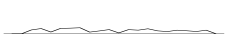
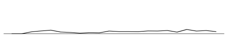
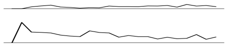
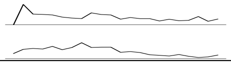
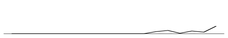
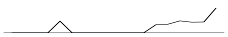
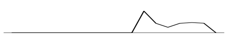
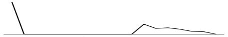
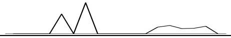
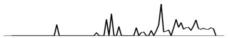
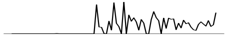
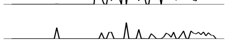
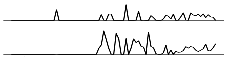
Peer-to-Peer network layer. We find 10 peer-to-peer network layer topics in our dataset, namely: ① *Network*, ② *Storage* for Bitcoin, ③ *Storage* for Ethereum, ④ *Network*, ⑤ *Channel (Setup)*, ⑥ *Storage*, ⑦ *Channel (Runtime)*, ⑧ *Fabric CA*, ⑨ *Identity* for Hyperledger Fabric, ⑩ *Storage* for Corda. Example post is:

> *Hyperledger fabric channel creation problems*
> I'm trying to create network hyperledger on raspberry, to test it I began with basic-network downloaded from official hyperledger repository. I modified the docker-compose.yml file using the frbrkoala images version armv7l-1.0.1 from the official docker repo, but I have problem on the channel creation.
>
> Dominant Topic: *Hyperledger Fabric/Channel (Setup)*

Development tools. We find 3 development tools topics, namely: ① *Truffle* for Ethereum, ② *Composer (Playground)*, ③ *Composer (REST Server)* for Hyperledger Fabric. Example post is:

> *Testing ethereum events directly in solidity with truffle*
> I found the following question for testing event logging in truffle using javascript. But truffle also supports writing tests directly in solidity. However, I can't find any documentation for how to test event logging in solidity. Can somebody help me with this?
>
> Dominant Topic: *Ethereum/Truffle*

TABLE 9
Trends of relative impact of topics in the studied blockchain platforms. For the trendlines, Y axes indicate the relative impact of topics; X axes correspond to the time span of a particular blockchain platform on Stack Exchange, i.e., May 2011 - December 2017 for Bitcoin, May 2015 - December 2017 for Ethereum, March 2016 - December 2017 for Hyperledger Fabric, July 2016 - December 2017 for Corda, and February 2011 - December 2017 for Other.

Topic Name	Trend (Bonferroni Corrected P-value)	Trendline
Bitcoin (K=10)		
Address	↑ (0.032)	
Storage	↑ (0.011)	
Transaction	↑ (9.7e-08)	
Wallet (Electrum)	↑ (0.0034)	
Wallet (blockchain.info)	↑ (0.00021)	
Client	– (1)	
Web API	– (1)	
Cryptocurrency	↓ (0.011)	
Mining	↓ (9.7e-08)	
Network	– (0.064)	
Ethereum (K=10)		
Smart Contract	– (0.21)	
Wallet	– (0.21)	
Account	– (1)	
Storage	– (1)	
Client	– (1)	
Solidity	– (1)	
Cryptocurrency	– (1)	
Transaction	– (1)	
Truffle	– (0.77)	
DAO	– (1)	
Hyperledger Fabric (K=15)		
Composer (REST Server)	– (0.98)	
Identity	↑ (0.0073)	
Channel (Runtime)	– (0.98)	
Channel (Setup)	– (0.98)	
Orderer	– (0.18)	
Storage	– (0.59)	
Composer (Playground)	– (1)	
Chaincode (Programming)	– (1)	
Fabric CA	– (1)	
Java SDK	– (1)	
Network	– (1)	
Peer (Command Line)	– (1)	
Transaction	– (1)	
Peer (Runtime)	– (0.18)	
Chaincode (Query/Deploy/Invoke)	– (0.18)	
Corda (K=5)		
Client	– (0.16)	
Transaction	– (1)	
CorDapp	– (0.16)	
Storage	– (0.63)	
State	– (1)	
Other (K=5)		
Hash Functions	↑ (2.2e-06)	
PHP Library for Coinbase API	↑ (3.1e-07)	
Python Library for Coinbase API	↑ (0.00026)	
Node.js Library for Coinbase API	↑ (8.7e-06)	
Coinbase Transaction	↑ (0.0021)