LEDGER
ledgerjournal.org

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

An Invitational Research Article from the ChainScience 2024 Conference

Investigating Similarities Across Decentralized Finance (DeFi) Services

Junliang Luo,* Stefan Kitzler,† Pietro Saggese‡

Abstract. We explore the adoption of graph representation learning (GRL) algorithms to investigate similarities across services offered by Decentralized Finance (DeFi) protocols. Following existing literature, we use Ethereum transaction data to identify the DeFi *building blocks*. These are sets of protocol-specific smart contracts that, similarly to "financial LEGO bricks", are utilized in combination within single transactions and encapsulate the logic to conduct specific financial services such as swapping or lending cryptoassets. We propose a method to categorize these blocks into clusters based on their smart contract attributes and the graph structure of their smart contract calls. We employ GRL to create embedding vectors from building blocks and agglomerative models for clustering them. To evaluate whether they are effectively grouped in clusters of similar functionalities, we associate them with eight financial functionality categories and use this information as the target label. We find that in the best-case scenario purity reaches .888. We use additional information to associate the building blocks with protocol-specific target labels, obtaining comparable purity (.864) but higher V-Measure (.571) and discuss plausible explanations for this difference. In summary, this method helps categorize existing financial products offered by DeFi protocols, and can effectively automatize the detection of similar DeFi services, especially within protocols.

1. Introduction

Decentralized Finance (DeFi) refers to a novel financial paradigm that leverages self-executing code deployed on top of Distributed Ledger Technologies (DLTs) known as smart contracts to provide financial functionalities within a decentralized framework. Thereby, it eliminates the need for intermediary entities like centralized financial institutions for transaction settlement.

The DeFi ecosystem is a thriving environment for financial innovation and the conception of new financial products.[1] Automated market-making, for instance, is a mechanism that facilitates the decentralized trading of cryptoassets by replacing order books.[2] The interoperability of smart contracts enables the creation of "DeFi compositions", where financial services of several DeFi protocols are combined to offer novel, complex and deeply nested financial products.[3] The DeFi landscape is indeed evolving rapidly: permissionless DLTs are censorship-resistant and

* J. Luo (junliang.luo@mail.mcgill.ca) is a PhD candidate in the School of Computer Science at McGill University, Montréal, Canada.
† S. Kitzler (kitzler@csh.ac.at) is a PhD candidate in the Complexity Science Hub, Vienna, Austria, and Junior Scientist at AIT Austrian Institute of Technology, Austria.
‡ P. Saggese (pietro.saggese@imtlucca.it) is an Assistant Professor at IMT School for Advanced Studies, Lucca, Italy.

their open-source design enables everyone to create new financial products and protocols. To date, it is hard to keep track of all existing DeFi projects and the financial services they offer. Notwithstanding this, such services often overlap in scope and purpose: most of them enable the lending and trading of cryptoassets, or devise yield-bearing strategies to reward liquidity provision. When protocol developers deploy newer versions of their projects, for interoperability purposes it is often essential that the newly deployed smart contracts are compatible with previous versions. It is also well known that code reuse is a practice conducted by protocol developers; as an example, the protocol SushiSwap is a fork of Uniswap.[4] Therefore, it is not uncommon that the implemented functionalities are similar if not identical across protocols. More generally, DeFi services reproduce in a decentralized context the functional logic of established "traditional finance counterparts". One could expect that the logic of such functions shares some fundamental characteristics. In other words, it is likely that different implementations of similar financial services share similar attributes and structure.

Recent research has disentangled the different financial services produced by DeFi protocols and has identified fundamental sets of protocol-specific smart contracts that, in combination, encapsulate the logic to conduct specific financial functionalities, such as swapping cryptoassets, or lending and borrowing them, calling them the *building blocks* of DeFi.[5] These building blocks are defined univocally by their shape and structure, which is modelled as a tree-like structure of interacting smart contracts. However, previous research did not investigate in detail the similarities and differences across building blocks, and to the best of our knowledge, no other studies have investigated in depth the entire space of such DeFi services.[6]

For the aforementioned reasons, we believe it is important to conduct a more detailed analysis of the various financial functions present within the DeFi ecosystem. On the one side, the growth without oversight of the DeFi ecosystem serves as a motivation for studies exploring solutions to automate the categorization of DeFi services; on the other side, evidence of similitudes on a technical level and of code reuse motivates studies that aim at investigating such similarities.

In this paper, we exploit machine learning algorithms to investigate how well it is possible to categorize building blocks in clusters of similar financial functionalities, based on their graph structure and attributes. We also aim to explore whether the financial functionalities of building blocks can be identified by their location within an embedding space or their proximity to certain other building blocks. Finally, we aim to understand the specific common design patterns that lead to the formation of these clusters and analyze imperfect classifications within them.

To answer these questions, we first replicate existing methods to obtain a set of building blocks encapsulating the main financial functions of DeFi.[5] To assess the similarity between these building blocks, we produce a similarity embedding space by applying graph-level representation learning and then exploit agglomerative clustering models on building blocks. We obtain four different specifications by associating various features to the nodes that compose the building blocks. Next, to evaluate how financial functionalities are grouped within such space, and to analyze whether distinct clusters represent particular financial operations, we gather additional information indicating the financial functionality category or the protocol they are associated with, and use it as the target label to compute a number of metrics such as homogeneity, completeness, V-Measure, and purity.

When evaluating the results using the "financial functionality category" label, we find that the outcomes yielded the highest value among all specifications for purity with .888, but a relatively

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

low V-Measure (.239). When evaluating the results using the "protocol" label, we find that the values for purity are comparable (.864 for the best-performing specification), and higher V-Measure (.571). To explain the difference in performance, we investigate more closely the common patterns within protocol-specific building blocks and look for plausible explanations. Finally, we identify protocol-specific patterns, that are re-used across them in different financial functionalities, which likely explain higher values for the clustering evaluated on the protocol target labels, compared to the financial functionalities.

The paper is structured as follows: Section 2 describes the concepts of building blocks, DeFi compositions, and the related literature; Section 3 describes the data and the methodology, while Section 4 shows results; finally, Section 5 reports discussion and concluding remarks. Data and code are available at: `https://github.com/JunLLuo/DeFi-similarity`.

2. Background

2.1. Decentralized Finance & DeFi Protocols—DeFi aims for open access for its users and provides a decentralized ecosystem that does not need intermediaries such as financial institutions to settle transactions. To date, Ethereum is the main blockchain for DeFi in terms of Total Value Locked (TVL).[7] In contrast to UTXO-based cryptocurrencies, such as Bitcoin, the Ethereum Virtual Machine (EVM) enables the use of Contract Accounts (CAs), also known as smart contracts, *i.e.*, software programs deployed on a blockchain. In contrast to Externally Owned Accounts (EOAs), CAs contain program code and, once deployed on the blockchain in question, methods (functions) can be called and the implemented logic will then be executed and computed. The most popular implementation of CAs is cryptoassets (later also just assets), representing real-world assets or rights on the blockchain. Prominent examples are stablecoins, such as Tether or USDC, whose value is pegged to the US dollar. DeFi can be thought of as an entire ecosystem of financial services for cryptoassets. DeFi protocols are implemented through CAs on the blockchain and provide financial functionalities, such as decentralized exchanges (DEX) or lending, to the end users. Many services automate their continuous code launch, creating factory-deployed (FD) contracts. Previous literature has extensively studied DeFi protocols, especially DEXs (Xu *et al.* (2023), Lehar & Parlour (2021), Heimbach *et al.* (2022))[2,8,9] using Automated Market Makers (AMM) (Fritsch *et al.* (2022),[10] Lending (Heimbach *et al.* (2023), Sun (2023), Xu & Vadgama (2022))[11–13] or Derivatives (Xiong *et al.* (2023)[14] protocols.

2.2. DeFi Compositions—Ethereum's flexibility allows CAs to be called by other CAs, therefore enabling smart contract interoperability. Consequently, entire DeFi protocols can leverage financial functionalities offered by other protocols and stack them similarly to "financial LEGO bricks".[15] Such a combination of multiple DeFi protocols, also known as DeFi compositions, might be beneficial for automating or providing more sophisticated actions. For instance, an aggregator protocol P_i can be used to determine the DEX protocol P_j with the best available price and execute the swap from Asset A to Asset B within the same, single transaction. Intuitively, this is a DeFi composition between P_i and P_j. Following a more thorough definition, *a DeFi composition within a single transaction is a coordinated combination of multiple Contract Accounts, associated with different DeFi protocols, where the CAs interact directly, or are linked by a common asset across the same transaction, to execute financial operations.*[3]

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

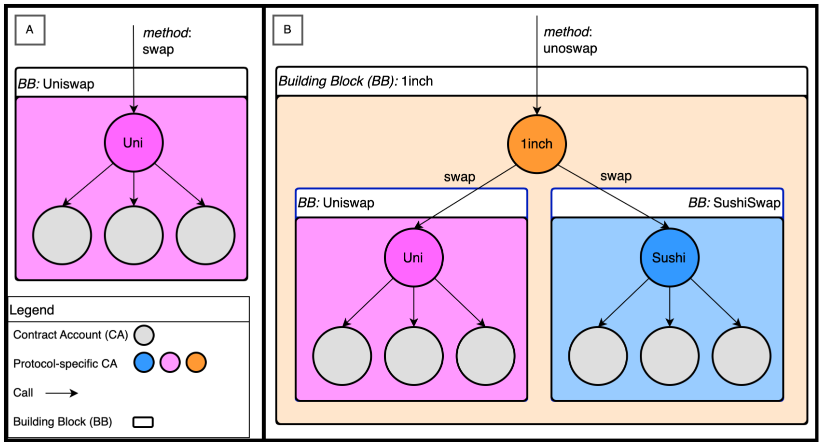

Fig. 1. Building blocks examples from *Uniswap* and *1inch*. Nodes represent account addresses and links are the calls of other accounts methods. Tree-structured transactions reveal the nested structure of composed DeFi services.

2.3. Conceptualization of Building Blocks—Whilst interoperability increases the potentialities of DeFi, it also adds a level of complexity to the system. Previous research investigated DeFi transactions in Ethereum in order to identify DeFi compositions and disentangle them, by proposing an algorithm that detects sets of smart contracts that, in combination, encode the logic of fundamental financial functions, also called building blocks.[5]

Building blocks are functional units of DeFi protocols that can be derived from the call structure of single transactions and are constructed as follows. Beginning with an EOA as the initiator of the transaction, the call of a CA can lead to subsequent internal transactions, *i.e.* calls of other CAs' methods, and ultimately culminate in a cascading, tree-like transaction structure. Building blocks can be extracted as sub-trees, where the root node (*i.e.*, the top node of the tree structure) is a protocol-specific CA with no incoming but at least one outgoing link. Furthermore, a building block may also encompass additional sub-trees, essentially forming nested structures of blocks. Figure 1 provides graphic support to understand this concept. It represents two frequently appearing building blocks from the popular DEX *Uniswap* (A) and one from the aggregator protocol *1inch* (B). Nodes represent account addresses and links represent the calls of CAs' methods. It illustrates the tree-like structure of building blocks and reveals the atomic *swap* block of *Uniswap* (A) to exchange assets. The more complex *1inch* building block (B) derives the best price across DEXs (when the method *unoswap* is called) and then executes two swaps by leveraging both building block (A) and an equivalent *SushiSwap* building block. Overall *1inch* is creating a DeFi composition, as intuitively described in Section 2.2.

2.4. Related Work—Compositions of DeFi protocols increase the ecosystem complexity, which makes systemic risk even harder to assess.[16–20] Early works were conducted for decomposing other elements of DeFi but not across protocols. Moin *et al.* (2019) decomposed the design of stablecoins into various component design elements and discussed the strengths and drawbacks.[21] Tolmach *et al.* (2021) specified systems compositions in automated market makers (AMMs) decentralized exchanges.[22] Von Wachter *et al.* (2021) described the "composed" derivatives of assets, in other contexts also known as "wrapped" tokens, and showed that the complexity of

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

wrapping operations has been peaking in the third quarter of 2020, a period often referred to as "DeFi Summer".[23] Existing online tools, such as DeFiLlama, provide categories of protocols, but do not necessarily distinguish among financial functionalities offered.

Other related studies are various machine learning-based analyses built upon extracting features of blockchain transactions to reveal the insights of the transactions in the blockchain blocks. Methods were proposed to process the input transaction graphs to construct desired sub-structure graphs or newly derived graphs for further analysis. Weber *et al.* (2019) explored the potential of Graph Convolutional Networks (GCN) for anti-money laundering (AML) in Bitcoin, contributing a publicly available labelled dataset of Bitcoin transactions for financial forensics.[24] Li *et al.* (2022) introduced a graph neural network-based model that incorporates temporal transaction data for the identification of phishing scams on the Ethereum network.[25] Pocher *et al.* (2023) demonstrated that employing Graph Convolutional Networks (GCN) and Graph Attention Networks (GAT) for modelling blockchain transactions as complex networks enhances the detection of AML and Countering the Financing of Terrorism (CFT) anomalies.[26] Han *et al.* (2023) developed a multi-layer graph neural network-based model for temporal transaction anomaly detection within Ethereum's multi-token transaction networks.[27] These studies leverage graph learning for transaction pattern recognition; yet, they lack a specific focus on smart contract interoperability and structural similarities across DeFi protocols, which our work aims to investigate.

3. Building Block Embedding

This section describes the dataset and the methodology devised to produce embeddings for DeFi building blocks. To uncover functional similarities and recurring interaction patterns, we first produce an embedding space where each building block is mapped to a high-dimensional vector by a graph representation algorithm; in this procedure, we assign various features to each node (*i.e.*, smart contract) of each building block. Similarity searches are conducted based on their distance within the space. We utilize two different target labels for evaluating the performance of our approaches in grouping DeFi building blocks into clusters based on their similarity.

3.1. Dataset and Sources Description—Following the approach introduced in Kitzler *et al.*,[5] building blocks are extracted utilizing all Ethereum transactions involving 23 DeFi protocols and their contracts from January 1, 2021 (block 11,565,019) to August 5, 2021 (block 12,964,999).[28] The algorithm extracts the building blocks by mapping the transactions into edge-induced subtrees, subsequently hashed based on a combination of their execution order, target nodes' outdegrees, and associated method. Each building block in the dataset has fields including the executed method's name, subtraces that outline the tree structure and participating addresses of the transactions within a building block, and a count that reflects the frequency of the building block's occurrence. The dataset we analyze consists of the top 10,000 building blocks identified using the building block extraction algorithm and ranked by count.

3.2. Methods of Producing Embedding—Given the building block tree-shaped structure, and since each node represents a smart contract with distinct features, the data are well-suited to apply graph representation learning to obtain graph-level embedding.[6] The method we employ, graph2vec,[29] is based on the Weisfeiler-Lehman (WL) method and was subsequently applied in performing graph isomorphism testing.[30,31] The graph2vec algorithms leveraging the WL

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

subtree kernel measure graph similarity quantitatively based on the commonality of their subgraph components, asserting that graphs share a higher number of common subgraph components and exhibit higher similarity. We apply graph2vec to generate a graph-level embedding space, yielding a singular vector for each building block. The embedding vectors abstract building block similarity relationships into a generalized representation, offering interpretability and predictive utility when used as input for models like classifiers or clustering algorithms.

3.3. Node Features—In graph2vec, the differentiation of the building block node features directly influences the characterization of the subgraphs, consequently affecting how closely graphs are positioned in the embedding space; graphs containing similar node features in addition to similar subgraphs are placed closer to each other. With graph2vec, we can provide node features or leave the field empty during the embedding vector generation process. We utilize various node features, detailed below, to investigate how they affect subgraph similarity in the embedding space.

3.3.1. None—For the baseline setting, no features are assigned to nodes. In this case, graph2vec will assign the node degree by default. The examination of the embeddings is based only on the building block graph structure without the influence of node-specific information.

3.3.2. 3-Class—We assigned features to the building block nodes following the distinction across simplified contract types described in the literature and reported in Section 2: factory deployed contracts (*i.e.*, contracts that generate other contracts), assets, and other contracts.[5]

3.3.3. Signatures Selectors—Each node N_i in a building block represents a contract address and therefore contains a list of functions. A function selector can be produced for each function, which refers to the first 4 bytes (8 characters) of the Keccak-256 (SHA3) hash of a function's signature, *i.e.*, the name of the function and its input argument types.[32] We utilized the signature extraction tools suggested by Di Angelo *et al.* for obtaining the selectors, representing the signatures, for every node within each building block across the entire building block dataset.[33,34] For each node, we generated a list of function selectors and assigned a unique marker to identical lists, utilizing this marker as the feature for the node.

3.3.4. Signatures Group—A potential limit of the *signatures* feature is that two contracts with mostly identical functions but minor variations would be assigned a different marker indicating distinct node features in the above function selectors feature. To precisely represent a smart contract node features by its functionality regardless of minor functional differences, we categorize contracts into distinct groups by evaluating the pairwise similarities of function selectors using the Jaccard metric (comparing the overlap between two sets of contract functions); *e.g.*, ERC20 contracts, despite minor variations, share common functions like transfer, approve, and balanceOf, resulting in high Jaccard scores and same grouping. Each building block will be assigned a marker indicating the function selector (representing the signature) group as its node feature.

3.4. Building Block Labels—To assess the performance of the embedding vectors in tasks that utilize these embeddings as input, we employed two sources as the target label: *Protocol* and *Financial Functional Category (FFC)*. To employ *Protocol* as the target label, the root node's protocol serves as the target label for the building block, meanwhile to use *Financial Functional Category* the process is slightly more complex.

To use FFC, we assign each building block to one of eight categories representing its financial functionality based on its root method name using regular expression patterns with the keywords

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

Table 1. Financial Functionality Category and the associated signature keywords.

Financial functionality category	Keywords	Count
Swap	‘swap’, ‘exchange’	4950
Lock Capital	‘deposit’, ‘add AND liquidity’, ‘staking’, ‘stake NOT unstake’, ‘lock NOT unlock NOT block’, ‘lend’, ‘collateralize’	550
Redeem or Withdraw	‘withdraw’, ‘remove AND liquidity’, ‘unstake’, ‘unstaking’, ‘unlock’	512
Borrow	‘borrow’	139
Get Interest or Rewards	‘(get OR claim) AND (reward OR fee)’, ‘harvest’, ‘earn’	129
Repay	‘repay’	36
Governance	‘vote’	16
Liquidate	‘liquidate’, ‘liquidation’	2
Others	-	3666

detailed below in Table 1. Regular expressions are used to search for specific keywords within the root method names of the building blocks. For instance, if a root method name contains substrings such as “deposit” or “lend”, it is categorized under the “Lock Capital” action. The presence of terms negated by “NOT” qualifiers, such as in “stake NOT unstake”, makes a method only categorized as “Lock Capital” if it involves “stake” without simultaneous involving “unstake”.

We note that the information used for the building block target label *Financial Functionality Category* differs from that used as node feature for the *Signatures Selectors* and the *Signatures Group*; indeed, the former uses information from the name of the function invoked only, while the latter two use data of *all* functions of a contract, included their arguments. Moreover, for the target label *Protocol*, we have labels for all 10,000 building blocks. For the *Financial Functionality Category* label, instead, not all building blocks contained one of the regular expressions defined in Table 1; these were categorized as “other” and excluded from the evaluation. We filtered out single-node building blocks, as they lack meaningful structural information for clustering and analysis purposes. Further details on the target labels are reported in Tables 3 and 4 in the Appendix.

4. Clustering Analysis

In this section, we describe the analysis devised to assess how effectively our embeddings categorize DeFi building blocks into clusters that reflect their financial functionalities, or other information such as the protocol they are associated with.

The procedure is presented in the workflow formulated in Algorithm 1. We conducted the analyses using each node feature $\mathscr{F}_{B_i}$ (see Section 3.3) and the target building block labels $\mathscr{L}_{B_i}$ (see Section 3.4), as described in line 3. We applied the graph2vec algorithm (line 4-8) with an embedding dimension of $\mu = 128$, a learning rate $\gamma = 0.05$, and $e = 100$ epochs to all the 10,000

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

Algorithm 1: Building Block Clustering

Data: A set of building block graphs $\mathscr{B} = \{B_1, B_2, ..., B_n\}$. Parameters: dimension μ, number of epochs e, learning rate γ. Clustering distance threshold: δ

1 . **begin**
2   **foreach** *building block* B_i *in* $\mathscr{B}$ **do**
3   Assign each node in B_i with the feature $\mathscr{F}_{B_i}$, and a set of labels $\mathscr{L}_{B_i}$
4   Initialize matrix $\Phi \in \mathbb{R}^{|\mathscr{B}| \times \mu}$ for building block embeddings
5   **for** *epoch* $e = 1$ **to** E **do**
6   Shuffle $\mathscr{B}$
7   **foreach** *building block* B_i *in* $\mathscr{B}$ **do**
8   Update $\Phi(B_i)$ using graph learning model (B_i, $\mathscr{F}_{B_i}$, μ, γ)
9   $\mathscr{C}, \mathscr{L}_{\text{cluster}} \leftarrow$ agglomerative_clustering(Φ, δ)
10   Assign predicted cluster $\mathscr{C}$ to each B_i
11   Evaluation: Calculate Homogeneity (H), Completeness (C), V-measure (V) and Purity (P) using $\mathscr{C}$ and labels $\mathscr{L}_{B_i}$.

building blocks. We used a threshold of 1.5 for Jaccard Ward distance when producing *Signatures Group* node features. After we obtained an embedding vector for each building block, we applied agglomerative clustering on all the embeddings.[35] To determine the optimal clustering distance threshold δ (line 9), we examined values within the range of [0.6, 1]. The value in this range that yielded the highest V-measure was chosen to have an optimal balance between homogeneity (to which extent each cluster contains only building blocks of a single target label) and completeness (to which extent all building blocks of a given target label are assigned to the same cluster), calculated by their harmonic mean. V-Measure and Normalized Mutual Information (NMI) with the arithmetic mean yield the same mathematical outcome since both ultimately measure the ratio of shared information (MI) relative to the total entropy.[36]

4.1. Results—We evaluate the clustering performance by computing the homogeneity, completeness, V-measure, and purity over both target labels and the four node features defined in Section 3. Table 2 reports the results of our analyses. We highlight in gray the best-performing specifications. Interestingly, we find that in all cases save one the node feature *Signatures Group* yields the best results. This is in line with our expectations since this feature is the most advanced and captures best the characteristics of the building blocks; the more information incorporated into the features, the better the results become.

We first focus on the *Financial Functionality Category* target label. In this specification, the outcomes yielded the highest value among the feature sets for purity with .888 (using *Signatures Group* as node feature). To further interpret these results, we reduce the embedding space into two dimensions using t-SNE method.[37] Figure 2 shows the embedding space of building blocks using *Signatures Group* as the node feature and *Protocol* as the label. In the figure, each dot represents a building block's embedding vector, with dimension reduced to 2D using t-SNE and its colour indicating the corresponding building block label. We find that building blocks associated with swapping functions, which are the vast majority of building blocks, are clustered close to each other. On the other side, completeness and V-measure are relatively low. This could be caused by clusters, such as "Redeem or Withdraw" and "Lock Capital", being not well separated. However,

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

this shows an interesting pattern, as the two functionalities are actually reciprocal to each other. In summary, the results suggest that the clustering categorizes DeFi building blocks into clusters where a substantial portion aligns with their distinct financial functionalities.

Table 2. Clustering results for building blocks with combinations of node features and building block labels. The best results for each target label are highlighted through gray shading, indicating that the *Signature Group* node feature produced the optimal clustering outcome evaluated by both target labels.

Target Label	Node Feature	δ	Cluster	Homogeneity	Completeness	V-measure	Purity
Protocol	None	0.64	294	0.371	0.202	0.262	0.582
	3-class	0.61	371	0.596	0.284	0.385	0.700
	Signatures Selectors	0.70	223	0.822	0.436	0.570	0.860
	Signatures Group	0.64	251	0.838	0.432	0.571	0.864
Financial Functionality Category	None	0.66	279	0.463	0.090	0.160	0.849
	3-class	0.70	280	0.594	0.120	0.200	0.856
	Signatures Selectors	0.60	283	0.689	0.135	0.225	0.887
	Signatures Group	0.62	265	0.706	0.144	0.239	0.888

Next, we look at the *Protocol* target labels and investigate how the performance changes. As Table 2 shows, the best-performance purity of .864 is comparable to the one of FFC, whilst all other measures are relatively higher. Figure 3 illustrates the observed separation by visualizing the embedding space using t-SNE dimension reduction (and additionally using different marker shapes to differentiate similar colours). The formed clusters show clear overall separation based on protocol: building blocks within the same protocol tend to share more common interaction patterns, characterized by similar sub-graph structures and features, whilst they are distinct from building blocks of other protocols. Notably, we observe that building blocks from *Uniswap* and *SushiSwap* exhibit close proximity, indicating overlapping functionalities as expected, since *SushiSwap* is a fork of *Uniswap*. Moreover, the building blocks of *Convex Finance*, a yield optimizer for the *Curve* protocol allowing users to earn increased rewards on the Curve token CRV, are positioned closely to the *Curve* blocks, possibly indicating reciprocal interactions.[38] In conclusion, we find that the graph embedding method works better in separating building blocks associated with the same protocol in comparison with FFC.

4.2. Further Insights—When using FFC as the target label, we observed a more fragmented scenario in clusters (see Figure 2) in contrast to *Protocol* as target. To look for plausible explanations and provide a deeper understanding of the method's performance, we therefore investigate more closely such differences. In Figure 4, we report illustrative examples of building blocks associated with the same protocol, that either contain the entire subgraph of another (left) or share common subgraphs (right). Numbers inside nodes represent the IDs of the function signature group features. Common transaction patterns are highlighted in red. On the left, the building block "borrow" of the lending protocol *Compound* contains the entire graph of "exchangeRateCurrent", although the financial functionalities differ. This re-used subgraph suggests that there likely are internal mechanisms that are not specific to the financial functionality, but instead are protocol-specific patterns. Also, on the right illustration we find for

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

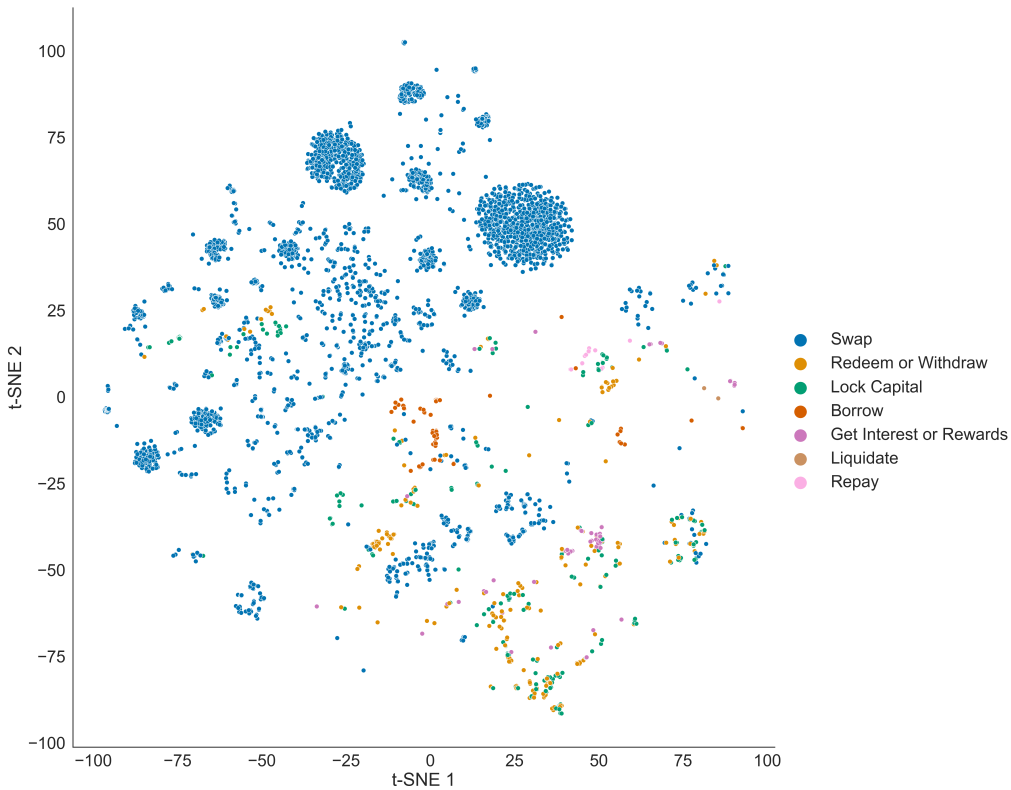

Fig. 2. Visualization of the embedding space of building blocks using *Signatures Group* as node feature and *Financial Functionality Category* as label. The building blocks within the Swap financial functionality category are well separated from the other categories and form multiple clusters. Building blocks of certain functionalities, such as "Redeem or Withdraw" and "Lock Capital" stay close, indicating overlapping characteristics despite different financial functionality.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

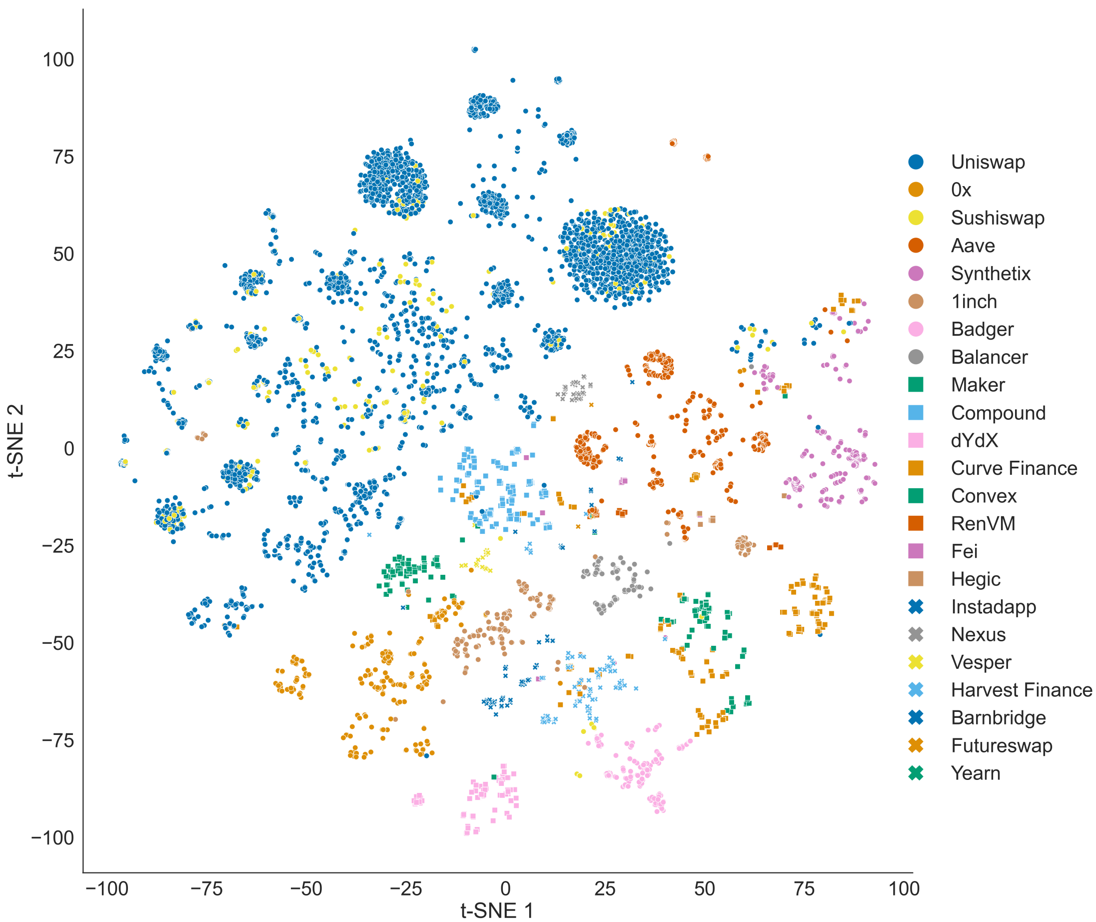

Fig. 3. Visualization of the embedding space of building blocks using *Signatures Group* as node feature and *Protocol* as label. Clusters of most protocols are markedly separated, with exceptions such as *Uniswap* and *SushiSwap*; this reflects well that the latter is a forked project of the former.

the lending protocol *Aave* an overlapping pattern, here with a common subgraph within "repay" and "borrow". Both examples are indicators for the existence of protocol-specific patterns, that are re-used across them, without necessarily being part of the financial functionality. These could be complementary CA-calls to offer the financial services, *e.g.* get token exchange rates, or serve as proxies, *i.e.*, many financial services are forwarded by the same instance. Also, the DeFi community widely supports the open source approach, which created synergies in standard code and library usage. Consequentially, these common subgraphs might explain higher measures of the protocol level clustering, compared to the financial functionalities.

5. Discussion and Conclusions

In this paper, we applied graph representation learning and agglomerative algorithms to cluster basic financial services offered by Decentralized Finance (DeFi) protocols, also called DeFi *building blocks*. We measured their similarity based on the subgraph structure of their corresponding DeFi transactions and by exploiting the smart contract attributes as node features. Using as

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

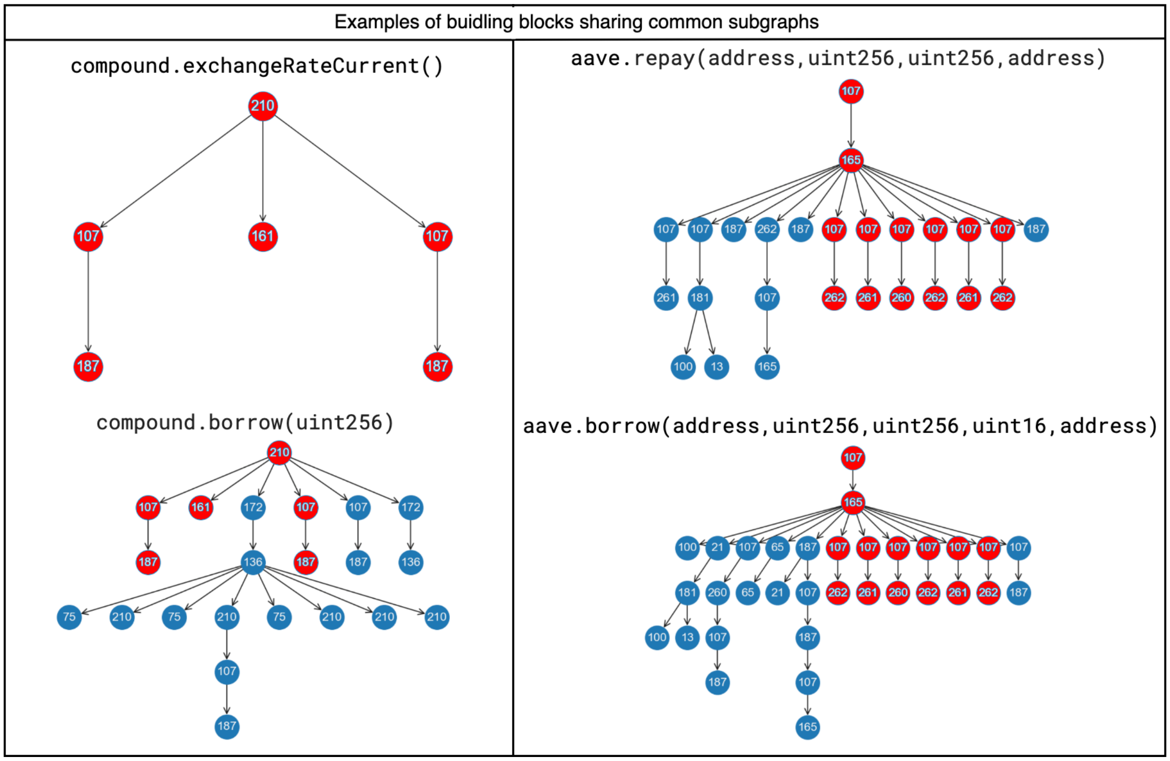

Fig. 4. Examples of building blocks with shared common subgraphs highlighted in red. Numeric values in nodes represent distinct *Signature Group* node features. Building blocks of same protocols, while differing in financial functionalities, can either contain the entire subgraph of another (left) or share common subgraphs (right), suggesting the presence of protocol-specific patterns reuse.

target labels i) information on the protocols and ii) the financial categories the building blocks are associated with, we can assess the effectiveness of our method and find that we are able to cluster building blocks associated with the same financial category with purity and V-measure respectively as high as .888 and .571.

Our method provides a framework for assessing the proximity of DeFi building blocks that offer similar functions. Several protocols exhibit a pattern characterized by protocol-specific clusters and a further grouping of within-protocol sub-clusters. Notably, we also identify relevant proximity relationships across protocols. The building blocks of *Curve* and *Convex Finance*—the latter being a yield optimizer for the former—are positioned close to each other; the building blocks of *Uniswap* and its forked protocol *SushiSwap* are in very near proximity and often overlap. Given the open source nature of many DeFi applications, forked projects share re-used code, resulting in similarities of building blocks. Our method could be useful to identify additional, novel proximal relationships when including the latest DeFi projects developed.

Our study also aids in systematically comparing different implementations of similar financial services, thus providing an overview of the most utilized DeFi building blocks by the category they are associated with. Building blocks facilitating swapping functions are well clustered

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

close to each other, and financial functions to withdraw or deposit funds, that are distinct yet encode a reciprocal, inverse functionality, are not clearly separated. As DeFi evolves, tracking all existing projects and services offered will be increasingly challenging. Our framework can help in comparing existing building blocks and in finding similar ones encoding common financial functionalities.

Similarly, our framework can enhance interoperability across protocols, acting as a support to effectively automatize and simplify the search of smart contracts that produce similar building blocks with comparable structure but providing slightly different functionalities. This, in combination with the use of compatible interfaces and function parameters, could facilitate the creation of novel DeFi compositions, created by replacing building blocks in use with alternative, similar ones identified with our setting. Even more, by knowing what building blocks exist (*i.e.*, their structure and interfaces), developers could create novel smart contracts specifically designed to facilitate interaction between existing building blocks, and thus create a completely novel array of compositions.

The devised methodology could benefit from implementing the following improvements. First, we only focus on transaction data and do not take additional blockchain-related event data into account to extract, *e.g.*, transfer patterns. Also, the dataset we used is limited in the time period analyzed and protocols included. Whilst the main purpose of the paper is to devise a novel approach to cluster similar building blocks, the extension to a longer time frame could provide further insights and also enable a temporal analysis. Finally, we currently focus only on a limited number of machine learning and graph representation algorithms. As a next step, we propose to complement them with additional analyses to cluster and categorize the DeFi functionalities.

Future work could investigate in greater detail the proximity relationships across protocols identified with our method and propose enhanced solutions to quantify them. For instance, while the proximity between *Uniswap* and *SushiSwap* might indicate the ability to identify forked projects, more specific testing and verification is necessary to substantiate such a hypothesis. To improve the clustering, future work should focus on handling protocol-specific transaction patterns. Also, a very promising approach to improve the methodology could entail the use of Large Language Models (LLMs), for instance, to extract smart contract information as a node feature. Such applications can be extended further, for example, to explore possible financial functionalities of building blocks emerging from new transactions, and predict their functionality based on their structure and attributes.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

Author Contributions

JL developed the algorithm and conducted the analyses; SK and PS extracted the data and conceptualized the research questions; all authors interpreted the results and contributed equally to the writing of the paper.

Funding and Acknowledgments

This work was supported in part by the Austrian security research program KIRAS of the Federal Ministry of Finance (BMF) under the project DeFiTrace, Grant 905300, and by the FFG BRIDGE project AMALFI, grant agreement 898883.

Notes and References

[1] Auer, R., Haslhofer, B., Kitzler, S., Saggese, P., Victor, F. "The Technology of Decentralized Finance (DeFi)." *Digital Finance* **6.1** 55–95 (2024) https://doi.org/10.1007/s42521-023-00088-8.

[2] Xu, J., Paruch, K., Cousaert, S., Feng, Y. "SoK: Decentralized Exchanges (DEX) with Automated Market Maker (AMM) Protocols." *ACM Computing Surveys* **55.11** 1–50 (2023) https://doi.org/10.1145/3570639.

[3] Kitzler, S., Victor, F., Saggese, P., Haslhofer, B. "A Systematic Investigation of DeFi Compositions in Ethereum." In S. Matsuo *et al.* (Ed.), *Financial Cryptography and Data Security. FC 2022 International Workshops. FC 2022. Lecture Notes in Computer Science*. **13412** 272–279 (2022) https://doi.org/10.1007/978-3-031-32415-4_18.

[4] Fan, S., Min, T., Wu, X., Wei, C. "Towards Understanding Governance Tokens in Liquidity Mining: A Case Study of Decentralized Exchanges." *World Wide Web* **26.3** 1181–1200 (2023) https://doi.org/10.1007/s11280-022-01077-4.

[5] Kitzler, S., Victor, F., Saggese, P., Haslhofer, B. "Disentangling Decentralized Finance (DeFi) Compositions." *ACM Transactions on the Web* **17.2** 1–26 (2023) https://doi.org/10.1145/3532857.

[6] Khan, A., Akcora, C. G. "Graph-Based Management and Mining of Blockchain Data." In *CIKM '22: Proceedings of the 31st ACM International Conference on Information & Knowledge Management* 5140–5143 (2022) https://doi.org/10.1145/3511808.3557502.

[7] Per https://defillama.com/chains.

[8] Lehar, A., Parlour, C. A. "Decentralized Exchange: The Uniswap Automated Market Maker." *SSRN* (2021) (Accessed 24 Jan 2025) https://dx.doi.org/10.2139/ssrn.3905316.

[9] Heimbach, L., Schertenleib, E., Wattenhofer, R. "Risks and Returns of Uniswap V3 Liquidity Providers." In *4th ACM Conference on Advances in Financial Technologies (AFT), Cambridge, Massachusetts, USA* 89–101 (2022) https://doi.org/10.1145/3558535.3559772.

[10] Fritsch, R., Käser, S., Wattenhofer, R. "The Economics of Automated Market Makers." In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies* 102–110 (2022) https://doi.org/10.1145/3558535.3559790.

[11] Heimbach, L., Schertenleib, E., Wattenhofer, R. "Short Squeeze in DeFi Lending Market: Decentralization in Jeopardy?" In A. Essex, *et al.* (Eds.), *Financial Cryptography and Data Security. FC 2023 International Workshops: Voting, CoDecFin, DeFi, WTSC, Bol, Brač, Croatia, May 5, 2023, Revised Selected Papers* 337–351 (2023) https://doi.org/10.1007/978-3-031-48806-1_22.

[12] Sun, X. "Liquidity Risks in Lending Protocols: Evidence from Aave Protocol." *arXiv* (2022) (Accessed 24 January 2025) https://doi.org/10.48550/arXiv.2206.11973.

[13] Xu, J., Vadgama, N. "From Banks to DeFi: the Evolution of the Lending Market." In N. Vadgama, J. Xu, P. Tasca (Eds.), *Enabling the Internet of Value: How Blockchain Connects Global Businesses* Cham: Springer 53–66 (2022) https://doi.org/10.1007/978-3-030-78184-2_6.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

[14] Xiong, X., Wang, Z., Chen, X., Knottenbelt, W., Huth, M. "Leverage Staking with Liquid Staking Derivatives (LSDs): Opportunities and Risks." *Cryptology ePrint Archive* Paper 2023/1842 (Accessed 25 January 2025) https://eprint.iacr.org/2023/1842.

[15] Popescu, A.-D. "Decentralized Finance (DeFi)–The Lego of Finance." *Social Sciences and Education Research Review* **7.1** 321–349 (2020) https://ideas.repec.org/a/edt/jsserr/v7y2020i1p321-348.html.

[16] Schär, F. "Decentralized Finance: On Blockchain- and Smart Contract-Based Financial Markets." *Federal Reserve Bank of St. Louis Review* **2** 153–74 (2021) https://doi.org/10.20955/r.103.153-74.

[17] Harvey, C. R., Ramachandran, A., Santoro, J. *DeFi and the Future of Finance*. Newark: John Wiley & Sons (2021).

[18] Kitzler, S., Saggese, P., Diem, C., Bernhard, H., Thurner, S. "Systemic Risk in Decentralized Finance (DeFi) - An Investigation of Smart Contract Interdependencies." In *Proceedings of the 11th International Conference on Complex Networks and their Applications* 233–235 (2022) https://dx.doi.org/10.5281/zenodo.7593062.

[19] Werner, S., Perez, D., Gudgeon, L., Klages-Mundt, A., Harz, D., Knottenbelt, W. "SoK: Decentralized Finance (DeFi)." In *AFT '22: Proceedings of the 4th ACM Conference on Advances in Financial Technologies* 30–46 (2022) https://doi.org/10.1145/3558535.3559780.

[20] Jensen, J. R., Ross, O. "Managing Risk in DeFi." *SSRN* (2020) (Accessed 24 January 2025) https://dx.doi.org/10.2139/ssrn.3745568.

[21] Moin, A., Sekniqi, K., Sirer, E. G. "SoK: A Classification Framework for Stablecoin Designs." In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers* Springer 174–197 (2020) https://doi.org/10.1007/978-3-030-51280-4_11.

[22] Tolmach, P., Li, Y., Lin, S.-W., Liu, Y. "Formal Analysis of Composable DeFi Protocols." In *Financial Cryptography and Data Security. FC 2021 International Workshops: CoDecFin, DeFi, VOTING, and WTSC, Virtual Event, March 5, 2021, Revised Selected Papers 25* 149–161 (2021) https://doi.org/10.1007/978-3-662-63958-0_1.

[23] von Wachter, V., Jensen, J. R., Ross, O. "Measuring Asset Composability as a Proxy for DeFi Integration." In *Financial Cryptography and Data Security. FC 2021 International Workshops: CoDecFin, DeFi, VOTING, and WTSC, Virtual Event, March 5, 2021, Revised Selected Papers 25* 109–114 (2021) https://doi.org/10.1007/978-3-662-63958-0_9.

[24] Weber, M., *et al.* "Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics." *arXiv* (2019) (Accessed 24 Janaury 2025) https://doi.org/10.48550/arXiv.1908.02591.

[25] Li, S., Gou, G., Liu, C., Hou, C., Li, Z., Xiong, G. "TTAGN: Temporal Transaction Aggregation Graph Network for Ethereum Phishing Scams Detection." In *WWW'22: Proceedings of the ACM Web Conference 2022* 661–669 (2022) https://doi.org/10.1145/3485447.3512226.

[26] Pocher, N., Zichichi, M., Merizzi, F., Shafiq, M. Z., Ferretti, S. "Detecting Anomalous Cryptocurrency Transactions: An AML/CFT Application of Machine Learning-Based Forensics." *Electronic Markets* **33** 37 (2023) https://doi.org/10.1007/s12525-023-00654-3.

[27] Han, B., Wei, Y., Wang, Q., Collibus, F. M. D., Tessone, C. J. "MT^{2}AD: Multi-Layer Temporal Transaction Anomaly Detection in Ethereum Networks with GNN." *Complex & Intelligent Systems* **10** 613–626 (2024) https://doi.org/10.1007/s40747-023-01126-z.

[28] Protocols are *Badger, Convex, Fei, Harvestfinance, RenVM, Vesper, Yearn, Barnbridge, dYdX, Futureswap, Hegic, Nexus, Synthetix, 0x, 1inch, Balancer, Curvefinance, SushiSwap, Uniswap, Aave, Compound, Instadapp,* and *Maker*. The choice of the protocols, their associated contracts, and the timeframe are conditional to the availability of sources utilized in Kitzler *et al.* (2022)[5]

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

[29] Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., Jaiswal, S. "graph2vec: Learning Distributed Representations of Graphs." *arXiv* (2017) (Accessed 24 January 2025) `https://doi.org/10.48550/arXiv.1707.05005`.

[30] Leman, A., Weisfeiler, B. "A Reduction of a Graph to a Canonical Form and an Algebra Arising During This Reduction." *Nauchno-Technicheskaya Informatsiya* **2.9** 12–16 (1968).

[31] Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., Borgwardt, K. M. "Weisfeiler-Lehman Graph Kernels." *Journal of Machine Learning Research* **12.77** 2539–2561 (2011) `http://jmlr.org/papers/v12/shervashidze11a.html`.

[32] See: `https://docs.soliditylang.org/en/v0.5.3/abi-spec.html#function-selector`.

[33] See: `https://github.com/gsalzer/ethutils/tree/main/doc/fourbytes` for the tools suggested by Di Angelo *et al.*.

[34] Di Angelo, M., Salzer, G. "Tokens, Types, and Standards: Identification and Utilization in Ethereum." In J. Xu, S. Schulte, P. Ruppel, A. Küpper, D. Jadav (Eds.), *2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)* 1–10 (2020) `https://doi.org/10.1109/DAPPS49028.2020.00001`.

[35] Murtagh, F., Legendre, P. "Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion?" *Journal of Classification* **31** 274–295 (2014) `https://doi.org/10.1007/s00357-014-9161-z`.

[36] Becker, H. "Identification and Characterization of Events in Social Media." (2011) PhD Thesis. New York: Columbia University (Accessed 24 January 2025) `http://www.cs.columbia.edu/~hila/hila-thesis-distributed.pdf`.

[37] Van der Maaten, L., Hinton, G. "Visualizing Data Using t-SNE." *Journal of Machine Learning Research* **9.86** 2579–2605 (2008) `http://jmlr.org/papers/v9/vandermaaten08a.html`.

[38] See: `https://docs.sushi.com/docs/Ecosystem/Whitepaper`, `https://docs.convexfinance.com/convexfinance`.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402

Appendix

Details on Building Block Labels—Tables 3 and 4 provide additional details on the target labels used to evaluate the clustering embeddings.

Table 3. Label Distribution within *Financial Functionality Category*. AF: After filtering.

Financial Functionality Category	**Count**	**AF**
Swap	4950	4930
Lock Capital	550	543
Redeem/Withdraw	512	506
Interest/Rewards	129	128
Borrow	139	139
Repay	36	36
Governance	16	8
Liquidate	2	2
Others	3666	3485
Total	10000	9877

Table 4. Label Distribution within *Protocol*. AF: After filtering.

Protocol	**Count**	**AF**	**Protocol**	**Count**	**AF**	**Protocol**	**Count**	**AF**
Uniswap	4475	4448	Badger	283	281	RenVM	61	58
Aave	795	795	dYdX	244	244	Vesper	44	42
0x	618	606	Harvest Finance	238	234	Hegic	36	34
SushiSwap	532	517	Balancer	207	202	Fei	30	25
Synthetix	500	497	Convex	197	195	Barnbridge	26	24
Curve Finance	495	489	Maker	194	182	Futureswap	11	11
Compound	424	420	Instadapp	148	148	Yearn	5	2
1inch	346	334	Nexus	91	89	Total	10000	9877

SSN 2379-5980 (online)
DOI 10.5195/LEDGER.2025.402