スマートポンジスキームにおける構文進化と検出性能に関する分析

大津 陸斗[1,a)]　Yin Minn Pa Pa[2]　吉岡 克成[2]　面 和成[1]

概要：
スマートコントラクト上で展開されるスマートポンジスキーム（SPS）は年々構文的に進化しており、従来の検出手法に対する有効性の低下が懸念される。最新の高性能な検出器 PonziGuard においても、構文進化やそれに伴う検出性能の影響についての分析は不十分である。本研究では、PonziGuard が検出した SPS を対象に、オペコード列に基づく構文クラスタリングや意味空間の可視化を行い、年代ごとの構文傾向を定量的に分析する。さらに、複数の分類器による年代別検出実験を通じて、構文の違いが検出性能に与える影響を定量的に評価する。我々の評価により、難読・委任型の構文に対して検出性能が著しく低下することが判明しており、構文多様性への適応が今後の SPS 検出器の設計において重要であることを示す。

キーワード：スマートコントラクト，ポンジスキーム, 機械学習， Fisher の正確確率検定

Investigating the Evolution of Syntax and Detection Effectiveness in Smart Ponzi Schemes

Rikuto Otsu[1,a)]　Yin Minn Pa Pa[2]　Katsunari Yoshioka[2]　Kazumasa Omote[1]

Abstract:
Smart Ponzi Schemes (SPS), which are deployed on smart contracts, have undergone increasing syntactic evolution over the years, raising concerns about the declining effectiveness of conventional detection methods. Even with PonziGuard, one of the latest high-performance detection systems, there has been insufficient analysis of syntactic changes and their impact on detection performance. This study focuses on SPS contracts detected by PonziGuard and conducts syntactic clustering based on opcode sequences as well as visualization in a semantic space to quantitatively analyze syntactic trends across different time periods. Furthermore, we evaluate the impact of syntactic differences on detection performance through time-series classification experiments using multiple classifiers. Our evaluation reveals a significant drop in detection performance against obfuscated and delegate-call-based contract structures, highlighting the importance of adapting to syntactic diversity in the design of future SPS detection systems.

Keywords: Smart contracts, Ponzi schemes, Machine Learning, Fisher's exact test

1. はじめに

近年、ブロックチェーン技術の発展や普及に伴い、金融や医療など様々な分野での応用が進んでいる。ブロックチェーンは分散型ネットワークを基盤として、取引データの透明性や耐改ざん性を実現する技術である。この技術により、第三者を介さずに安全かつ効率的な取引が可能となり、多くの業界で注目を集めている。また、分散型金融（DeFi）やトークンの発行、投票システムなど、幅広いアプリケーションの導入が進んでいる。その中でもスマートコントラクトは、ブロックチェーンアプリケーションの開

[1] 筑波大学
University of Tsukuba
[2] 横浜国立大学
Yokohama National University
[a)] s2420528@u.tsukuba.ac.jp

発で用いられる技術であり、幅広い分野での活用が期待されている。

しかし、ブロックチェーン技術やスマートコントラクトの普及に伴い、その匿名性や耐改ざん性を悪用する詐欺行為の増加が懸念されている。その代表的な詐欺の一つが、スマートポンジスキームである。スマートポンジスキーム（SPS）は、従来のポンジスキームをスマートコントラクト上で自動化した詐欺手法であり、投資家に高収益を約束しながら、新たな投資家から集めた資金を既存の投資家に配当として支払うモデルをとっている。被害が広がるほど損失額が膨大になり、深刻な経済的影響をもたらしている。そのため、SPS の早期検出と防止が急務となっている。

近年、SPS の検出手法として、機械学習を用いたアプローチを採用した研究が行われている。これらの手法は、従来のルールベースアプローチに比べ、パターン認識や異常検出において一定の成果を示している。現在、最も有効だと考えられている手法として、Liang らの研究 [1] があげられる。彼らは PonziGuard という手法を提案し、従来のオペコードの頻度やトランザクション履歴を用いた機械学習アプローチと比較して優れた検出性能を達成している。また、Ethereum Mainnet 上における評価では、805 件の SPS 検出など実データにおける有効性なども示している。

しかし、この手法にもいくつかの課題がある。第一に、実験に用いられたデータが最新の詐欺傾向、特に構文の多様化や難読化を十分に反映しているかは明らかでない。第二に、検出漏れに関する定量的評価が不十分であり、検出性能の全体像を把握するには限界がある。第三に、検出された SPS の変化が、SPS そのものの衰退によるものなのか、あるいは検出器の限界による見落としなのかといった点についても明確な説明がなされていない。

そこで、本研究では、構文的特徴の進化やその影響を定量的に分析し、同手法の適応限界を評価することで、将来的な検出器の性能向上や継続的なモデル改善に向けた課題と方向性を明らかにすることを目的とする。そのために、以下の 2 つの Research Question（RQ）を設定する。

RQ1: SPS コントラクトの構文的特徴には、年代に応じた変化が存在するか

RQ2: 上記の構文進化は、検出器の性能にどのような影響を及ぼすか

この 2 つの RQ のため、PonziGuard によって検出されたスマートポンジスキームとされるコントラクトを対象に、オペコードに基づく構文的特徴の変化を時系列分析した。そして、この変化が検出性能に与える影響を評価するため、複数の分類器を用いて年代別の検出性能を比較し、構文進化による影響や誤分類傾向を明らかにした。これにより、スマートポンジスキーム検出手法の設計において、構文的多様性への対応やモデルの継続的な更新の重要性を示した。

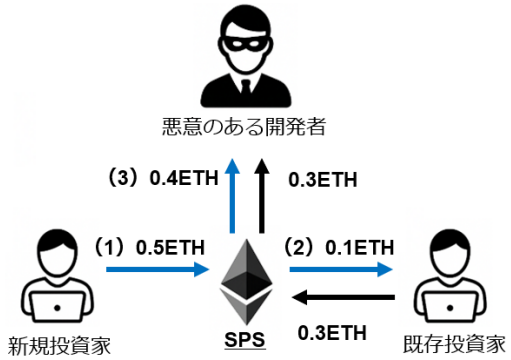

図 1: スマートポンジスキームのフロー（例）

2. 準備

2.1 スマートコントラクト

スマートコントラクトは、特定の条件が満たされた際に、自動で契約や取引を実行するプログラムであり、主にイーサリアムなどのブロックチェーン上で動作する。イーサリアム仮想マシン（EVM）は、スマートコントラクトの実行環境として機能し、開発者が安全で効率的なコードを構築できる基盤を提供している。スマートコントラクトの主な利点は、従来の契約で必要だった仲介者や第三者の関与を排除し、取引コストの削減や透明性を向上させる点である。また、スマートコントラクトは一度ブロックチェーン上に記録されると、後から変更や削除することができないため、高い耐改ざん性を持つ。この特性により、スマートコントラクトは信頼性が一層高まっている。

2.2 スマートポンジスキーム

スマートポンジスキームは、イーサリアム上で発見された詐欺手法であり、スマートコントラクトを利用して従来のポンジスキームを自動化している。ポンジスキームとは、高収益をうたい新規投資家を募り、その資金を既存投資家に配当する構造であり、参加が途絶えると資金が枯渇し破綻する。

図 1 のように、スマートポンジスキームでは、ユーザが投資プログラムを装ったコントラクトに資金を送金すると、その一部が既存参加者に再配当され、残りの多くが悪意のある作成者によって回収される。スマートコントラクトの使用によって、この過程が自動化され、透明かつ改ざん不可能なブロックチェーン上で実行される点が特徴である。

しかし、匿名性や永続性は、詐欺行為の隠蔽や被害回復の困難さを招いている。加えて、コードが公開されている点から信頼性に誤解を生み、ユーザが仕組みを理解しないまま参加する例も少なくない。

このような背景から、スマートポンジスキームは従来よりも変更不可能な点や被害規模の点で深刻なリスクを伴う。Bartoletti らの研究 [2] では、Ethereum 上のポンジスキー

ムに関する実態調査などを行い、その実装パターンや長期的な経済的影響の深刻さを明らかにしている。現在は、こうした詐欺の早期検出を目指し、機械学習などを用いた分析手法の研究が進められている。

3. 関連研究

機械学習を利用したスマートポンジスキーム検出においては、主に取引情報、コード情報、ランタイム情報を活用した機械学習アプローチが提案されている。以下に、それぞれの代表的な研究を紹介する。

3.1 取引情報を用いた検出手法

Hu らの研究 [3] では、Ethereum 上のスマートコントラクトにおける 10,000 件以上のトランザクションデータを分析し、行動パターンから 14 の特徴量を抽出して LSTM による分類を行っている。これにより、ゲームや金融、ギャンブルなどのコントラクトを高精度に識別できるほか、異常な動作をする不正コントラクトの検出も可能であることが示されている。

また、ポンジスキーム検出に特化した Chen らの手法 [4] と比較した結果、Hu らの手法はオペコードを用いないため精度は劣るが、汎用性が高く、不正検出以外の応用も可能である点が特徴である。

3.2 コード情報を用いた検出手法

スマートコントラクトのコード情報、特にオペコードの頻度分布を用いた検出手法は、多くの研究で有効性が示されている [4], [5], [6], [7]。Wang ら [5] は、オペコードとアカウント情報を LSTM に入力し、SMOTE（Synthetic Minority Over-sampling Technique）[8] による不均衡対策を施した PSD-OL（Ponzi Scheme Detection via Oversampling-based Long Short-Term Memory）を提案。JUMP 命令の多用など SPS 特有の特徴を捉え、従来手法より高精度な検出を実現している。一方で、詐欺手法の進化（コンセプトドリフト）への対応が今後の課題とされる。

Zheng らの研究 [7] では、取引情報に依存せず、スマートコントラクトのバイトコードと開発者情報を用いた MulCas（Multi-view Cascade Ensemble）が提案された。MulCas の特徴は、詐欺の実行を待たずにスマートコントラクトが作成された時点で早期検出可能な点にあり、被害の早期抑止に有効である。また、オペコードの頻度や N-gram などに加え、開発者ベクトルという新しい特徴量を活用し、有効性を示した。さらに、不均衡データへの対応や高い再現率・F1 スコアにより、従来手法を上回る性能を示している。

一方で、敵対的攻撃（Adversarial attacks）への脆弱性や、使用している XBlock データセット [9] の陳腐化が課題とされており、モデルの堅牢性向上とデータ更新の継続が今後の課題である。

3.3 ランタイム情報を用いた検出手法

Liang らの研究 [1] では、ランタイム情報を活用した検出手法である PonziGuard を提案している。特徴として、各関数のオペコード列を Doc2Vec でベクトル化し、スタック操作量・資金フロー・関数間呼び出しなどの実行時情報を組み合わせた CRBG（Contract Runtime Behavior Graph）を構築し、GNN（Graph Neural Network）で分類を行っている。

実験では、従来手法をすべての評価指標で上回り、Mainnet 上で 805 件の新規ポンジスキームを検出。うち 50 件のソースコードを確認した結果、すべてが詐欺であることから有効性を示した。被害総額は約 281,700 ETH と推定される。実行時情報の取得に伴う処理コスト（EVM の改造含む）は許容範囲とされる一方、fuzzing やグラフ構築を含めた全体処理時間は大きく、全件スキャンやリアルタイム検出にはスケーラビリティ上の課題があると考えられる。

3.4 既存手法における課題

関連研究を通じて、スマートポンジスキーム検出に関する既存研究には、以下の課題がある。

まず、データセットの更新性が不十分である。多くの研究で使用されている XBlock などのデータセット [9] は 2019 年以前の情報に基づいており、設計の多様化・難読化が進む近年のスキームには対応できていない可能性がある。また、各年のデータ数に偏りがあり、時系列評価の信頼性にも懸念がある。これらを解消するには、最新かつ網羅的なデータセットの構築と継続的な更新、検出モデルの適応性および性能向上が必要である。

次に、データの不均衡性が検出性能に与える影響である。ポンジスキームのラベル付きデータは依然として少なく、過学習やモデルの汎化性能の低下が懸念される。SMOTE のようなオーバーサンプリング手法や、Wang らの研究 [10] のように少数クラスの影響を受けにくい頑健なモデル開発が求められる。

さらに、敵対的攻撃への脆弱性も重要な課題である。特にコード情報に依存する手法では、無意味なオペコードを挿入することで検出を回避する攻撃（Adversarial Examples）が可能であり、検出精度の低下を招く恐れがある。

4. PonziGuard における課題

前節で述べたように、Liang らの研究 [1] は、ランタイム情報を用いた手法である PonziGuard を提案し、グラウンドトゥルースデータおよび Ethereum Mainnet 上で高い性能を示している。一方で、本研究では以下の課題が考えられる。

- **構文的傾向に関する定量的分析の欠如**：PonziGuard では検出された SPS の件数などの分布について簡単な言及があるものの、構文的特徴などの時系列的変化に

ついては定量分析はされていない。また、COVID-19 などの社会的背景と SPS の出現動向との関連が示唆されているものの、それらと構文設計との関係についての因果的・定量的な議論はなされていない。

- **検出性能に関する評価の限定性**：Mainnet 上で 805 件の SPS を検出し、そのうち 50 件の真陽性確認を行っているが、Recall や F1 スコアといった定量指標による漏れの評価は行われていない。また、構文傾向の変化による誤分類（False Negative）のリスクは検証されておらず、検出器の堅牢性について評価が不十分である。

これらの点から、最新の傾向や構造的変化に対する検出器の適応力や、評価指標の妥当性について慎重な検討が引き続き求められている。

そこで、スマートポンジスキームの検出手法の提案においては、検出対象の構文的特徴を把握し、その変化と検出性能との関係を明らかにすることが重要である。RQ1 では、構文的傾向の変化を時系列で定量的に明らかにすることを目的とし、RQ2 では、構文進化が検出性能に与える影響を分類器によって定量的に評価することを目的とする。

5. RQ1：SPS コントラクトの構文特徴における年代別評価

スマートポンジスキーム（SPS）の検出においては、構文の多様化や難読化といった構文進化が、既存の検出器の大きな課題となっている。特に、SPS の開発者は検出を回避する目的で構文を意図的に変更している可能性があり、それに伴う時系列的な構文変化を把握することは、検出器の限界を理解し今後の設計指針を立てる上で重要である。

本章では、PonziGuard[11] によって検出された SPS を対象に、オペコード列に基づくクラスタリングと統計的分析を通じて、構文傾向の時系列的変化を明らかにする。

5.1 データセット

本研究では、Liang らの研究成果に基づき、GitHub 上に公開されている PonziGuard により検出されたとされる 805 件のスマートポンジスキームのコントラクトアドレス [11] を使用する。各アドレスに対して、Etherscan API [12] を用いて、コントラクトの展開時刻（作成日時）やコード情報を取得し、分析可能な形式に整形する。なお、Etherscan 上では一部のコントラクトについて、ソースコードやバイトコードが公開されておらず取得できない場合がある。本研究ではそのようなケースに該当する 63 件を除外し、最終的に 742 件の SPS を分析対象とする。バイトコードについては、オペコード列に変換し、構造的特徴の抽出に利用する。

5.2 分析手法

本研究では、RQ1 の検証のために、スマートポンジスキーム（SPS）のオペコード列に基づいた構文的クラスタリングと、クラスタごとの特徴語（代表的命令語）の抽出を行う。分析は以下の 3 つのステップで構成される。

5.2.1 Step 1：ベクトル化とクラスタリング

前節で得られた 742 件のコントラクトのオペコード列を文書とみなし、TF-IDF によるベクトル化を実施する。これにより、各コントラクトをオペコードの出現頻度パターンに基づいて定量的に表現できるようになる。

その後、k-means クラスタリングを適用し、構文的に類似した SPS をクラスタごとに分類する。クラスタ数 k は Silhouette スコアの変化率を確認しつつ最適値を設定し、本研究では $k = 5$ とした。さらに、各年における SPS のクラスタ構成比を示すことで、年代別の傾向変化を分析する。

5.2.2 Step 2：クラスタ特徴語の定量的抽出

得られた各クラスタ C_i について、その構文的特徴を定量的に把握するため、クラスタ内とそれ以外（補集合 $\neg C_i$）のコントラクト群を比較対象とし、各命令語（オペコード）w の出現率差[*1]を算出する。出現率差 $\Delta(w, C_i)$ は以下の式で定義される。

$$\Delta(w, C_i) = \frac{n_{C_i}(w)}{N_{C_i}} - \frac{n_{\neg C_i}(w)}{N_{\neg C_i}} \tag{1}$$

ここで、$n_{C_i}(w)$ はクラスタ C_i に属するコントラクトのうち命令語 w を含む件数、N_{C_i} はクラスタ C_i の総コントラクト数、$n_{\neg C_i}(w)$ および $N_{\neg C_i}$ は補集合内での対応する値を表している。

この出現率差が統計的に有意であるかどうかを検定するために、命令語 w ごとに 2×2 の分割表を作成し、Fisher の正確確率検定[*2][13] を実施する。また、複数命令語に対する多重比較による偽陽性率の影響を抑えるため、Benjamini–Hochberg 法 [14] による FDR 補正を行い、補正後の p 値が 0.05 未満である命令語を統計的に有意なクラスタ特徴語と定義する。この処理をクラスタごとに繰り返し、各クラスタで出現率差 $\Delta(w, C_i)$ が大きく、かつ統計的に有意である上位 10 件の命令語を、そのクラスタの代表的な構文的特徴として抽出する。

なお、単純な出現頻度や TF-IDF 値のみを指標とした場合、頻出する一般的な命令語（例：`PUSH1`, `JUMPDEST` など）が上位に並ぶ傾向があり、クラスタ間の違いが埋もれる恐れがある。そのため、本研究では「クラスタ内で特に偏って出現する命令語」に着目した上記の手法を用いることで、クラスタごとの識別に有効な構文特徴をより明確に抽出している。

[*1] ある命令語（オペコード）について、特定クラスタ内とそれ以外のクラスタでの出現率の差を指す。本研究ではこの差が大きい命令語を特徴語として抽出する。

[*2] この検定は、2 つのカテゴリ変数間の有意な関係を検出する統計的手法である。本研究では、命令語の出現がクラスタ間で偏っているかを検定するために用いる。

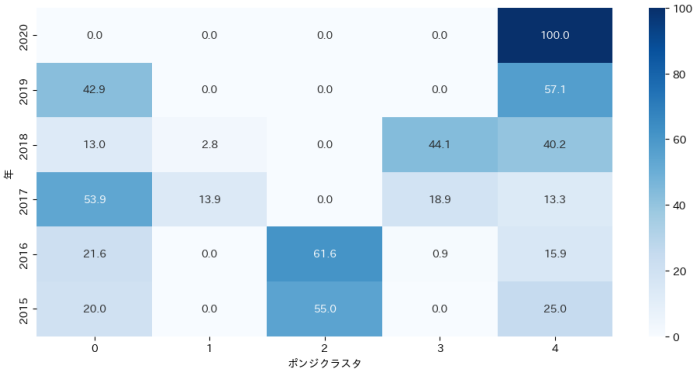

図 2: クラスタごとの年代別構成比率

5.2.3 Step 3：構文意味空間の可視化

クラスタの意味的類似性を視覚的に確認するため、各命令語を Word2Vec でベクトル化した上で、各コントラクトの平均ベクトルを算出し、コントラクト単位の意味ベクトルを構築する。その後、主成分分析（Principal Component Analysis, PCA）を用いて 2 次元に次元削減し、クラスタの意味的重心を 2 次元平面上にプロットする。

この分布の可視化により、クラスタ間の意味的な距離や類似性、または異質な構文傾向を持つクラスタの特定が可能となる。これにより、構文の出現傾向だけでなく、構文の意味的進化や乖離を視覚的に把握することができる。

5.3 結果

本節では、分析手法で示したクラスタリングおよび統計的分析に基づく構文特徴における年代別評価を示す。具体的には、(1) SPS のクラスタ構成比の時系列変化、(2) 各クラスタの統計的に有意な代表オペコード、(3) 意味空間上におけるクラスタ間の類似性の可視化、の 3 点に着目する。

5.3.1 クラスタ構成比の時系列的変化

図 2 は、2015 年から 2020 年までの各年における SPS のクラスタ構成比率を示している。2015〜2016 年には、Cluster 2 が全体の過半数を占めており、初期の SPS が特定の構文パターンに集中していたことが分かる。

一方、2017 年には構文パターンの分化が進み、Cluster 0（53.9%）、Cluster 1（13.9%）、Cluster 3（18.9%）などの構成比が増加するなど、クラスタ構成に大きな変化が生じている。その後、2018 年には Cluster 3（44.1%）や Cluster 4（40.2%）が台頭し、2020 年にはすべての検出 SPS が Cluster 4 に分類されるなど、構文的多様性が年を追って減少する傾向が確認された。特に注目すべきは、2019 年以降に Cluster 1 と Cluster 3 の構成比が 0%となっている点である。これらの結果は、時系列における構文進化と、何らかの検出限界または開発者の回避行動を示唆している。

5.3.2 クラスタごとの代表的命令語

表 1 は、各クラスタにおいて出現率差 $\Delta(w, C_i)$ が大きく、かつ Fisher の正確確率検定によって $p < 0.05$ の統計的有意性が確認された代表的オペコードを 5 件ずつ参考として示したものである。これらの命令語の出現傾向に基づき、各クラスタには以下のような構文的傾向が見られる。

表 1: 各クラスタにおける代表的オペコード

Cluster	Opcode	出現率差	p 値
0	SWAP6	0.470	< 0.001
	DUP14	0.357	< 0.001
	SWAP12	0.331	< 0.001
	TIMESTAMP	0.315	< 0.001
	LOG1	0.287	< 0.001
1	DELEGATECALL	0.970	< 0.001
	INVALID_0x6c	0.830	< 0.001
	UNKNOWN_0xd0	0.811	< 0.001
	PUSH19	0.772	< 0.001
	CALLDATACOPY	0.677	< 0.001
2	UNKNOWN_0xa9	0.522	< 0.001
	UNKNOWN_0xd	0.517	< 0.001
	UNKNOWN_0xec	0.514	< 0.001
	UNKNOWN_0xd9	0.509	< 0.001
	INVALID_0x64	0.492	< 0.001
3	PUSH29	0.550	< 0.001
	REVERT	0.536	< 0.001
	UNKNOWN_0xfe	0.474	< 0.001
	PUSH6	0.413	< 0.001
	LOG1	0.353	< 0.001
4	DUP10	0.373	< 0.001
	UNKNOWN_0xfe	0.335	< 0.001
	REVERT	0.333	< 0.001
	DUP7	0.308	< 0.001
	DUP8	0.306	< 0.001

- **Cluster 0（状態チェック型）**：swap や DUP に加え、BALANCE や TIMESTAMP など、外部状態の取得命令が多く、送金条件の検証に重点を置く構文。
- **Cluster 1（難読・委任実行型）**：DELEGATECALL や UNKNOWN_0xd0 などの未定義命令が多く、難読性・外部委任実行が特徴。コードの構造が不透明で、解析を困難にする設計が見られる。
- **Cluster 2（ガス制御・初期化型）**：CODECOPY や GASLIMIT、CODESIZE といった命令から、初期化コードの制御が中心的役割を担っており、コントラクト展開時に挙動を動的に構成する傾向。
- **Cluster 3（異常処理・偽装型）**：REVERT や LOG1、GAS 命令が目立ち、返金処理や異常終了を装うことでユーザを欺く意図が示唆される。
- **Cluster 4（制御構文型）**：DUP や PUSH などの基本命令を組み合わせた単純構文が中心で、抽象度が高く、多くの状況に応じた繰り返し処理や状態遷移を可能とする構成。

これらのクラスタは、単なる頻度ベースの分析では抽出しにくい構文的偏りを捉えており、時系列における出現傾

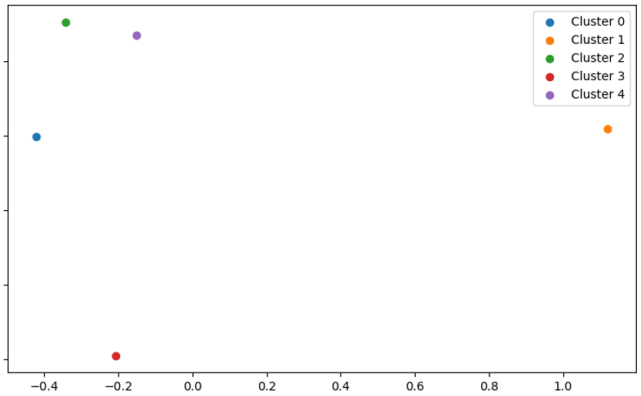

図 3: 各クラスタの意味的重心分布（PCA による次元削減）

向とあわせて、検出器の脆弱性を分析する上で有用である。

5.3.3 意味空間におけるクラスタの位置関係

図 3 は、各クラスタの構文的な意味的距離を補足的に検証するため、Word2Vec によって得られたベクトル表現を用い、各平均ベクトル（意味ベクトル）を算出し、クラスタごとの重心を PCA で 2 次元に可視化したものである。

この図からは、Cluster 1 および Cluster 3 が他のクラスタと大きく離れた位置に分布しており、意味的にも異質な構文パターンを有していることが確認された。特に、難読化命令や偽装処理を含む構文が他のクラスタと明確に異なるベクトル空間上の性質を持っている点は、検出器がこれらに対応しきれていない可能性を補強するものである。

一方、Cluster 0・2・4 は比較的近接しており、意味ベクトルの共通性が高いといえる。これらのクラスタは、構文の抽象度や処理目的が似ていると考えられ、既存の検出器が比較的対応しやすい構文パターンと示唆される。

6. RQ2：SPS コントラクトの構文特徴における検出性能

前章では、スマートポンジスキーム（SPS）の構文傾向が年代によって変化していることを示した。本章では、こうした構文進化が検出器の性能に与える影響を検証することを目的とし、分類モデルを用いた性能比較実験および誤分類パターンの分析を通じて、検出漏れや性能変化など構文進化との関係性を定量的に評価する。

6.1 データセット

本分析では、RQ1 と同様に PonziGuard により検出された 742 件の SPS を対象とし、コントラクトの展開時刻に基づいて 2015–2016 年、2017–2018 年、2019–2020 年の 3 期間に分割して用いる。

また、検出器の分類性能を評価するため、SPS でないコントラクトを対照群として使用する。これらは、Etherscan API [12] を用いてランダムに収集され、年代によるデータの偏りを避けるため、各年次の SPS と同数となるようにサンプリングする。

すべてのコントラクトはオペコード列に変換した上で、分類モデルにおける特徴量生成および学習・検証に利用する。これにより、構文的特徴に基づく分類器の性能が、構文進化によってどの程度変動するかを定量的に評価可能となる。

6.2 分析手法

本研究では、RQ2 の検証のために、構文進化が検出性能に与える影響を定量的に評価する分類実験を実施する。特に、構文的進化によって既存の検出器が False Negative（FN）を生じる傾向に着目し、以下のような分析手法を設計する。

6.2.1 Step 1：分類性能の年代別比較実験

構文的傾向の変化が分類性能に与える影響を明示的に把握するために、学習期間を 2015–2016 年に固定し、検証対象として 2015–16 年・2017–18 年・2019–20 年のデータセットを使用する。これにより、構文進化がモデルの凡化性能に及ぼす影響を比較可能とする。

分類モデルには、先行研究で用いられている Ridge、SVM、Random Forest、XGBoost、LightGBM の 5 種類を採用する。また、特徴量としては、オペコード列から得られる TF-IDF や Term Count、N-gram（bi-gram）、Ngram に基づいた TF-IDF、Word2Vec を用いる。これらの分類モデルと特徴量の組み合わせにより、各年代における分類性能（F1 スコア）の変化を定量的に評価する。

6.2.2 Step 2：誤分類分析による

分類性能が低下した場合に、誤分類の中でも特に False Negative（FN）が特定の構文クラスタに偏っていないかを分析する。これは、RQ1 で構築したクラスタ分類結果と組み合わせて実施する。

なお、False Negative に注目するのは、本分析でクラスタリングや特徴抽出の対象となるのが SPS に限定され、検出漏れ（FN）のみが構文的特徴の分析対象となるためである。また、False Positive に比べ、SPS の見逃しは実被害に直結するリスクが高く、重要な指標と考えられる。

特に、F1 スコアの変動が最も顕著だった分類器（本研究では XGBoost + TF-IDF の組み合わせ）において、2017–2018 年の検出失敗サンプル（FN）を対象とし、それらの構文クラスタの分布を集計・分析する。

この分析により、以下の点を検証する。

- 特定の構文クラスタに検出漏れが集中しているか
- クラスタの性質が誤分類傾向と関連しているか

これにより、構文的に異質な SPS パターンが検出器の見逃しとなっている可能性を補足的に明らかにし、今後の検出器設計における課題を提示する。

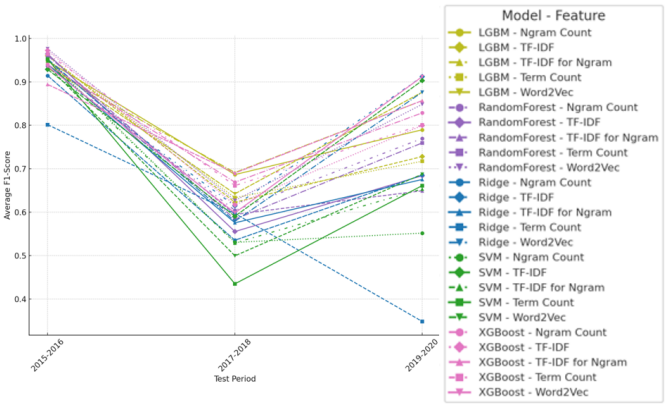

図 4: 各モデル・特徴量による年代別 F1 スコア

表 2: 2017–2018 年における誤分類（FN）のクラスタ分布

クラスタ	誤分類比率
Cluster 1	67.2%
Cluster 0	20.8%
Cluster 4	9.5%
Cluster 3	2.4%
Cluster 2	0.0%

6.3 結果

6.3.1 分類性能の年代別比較実験の結果

図 4 に示すように、2015–2016 年の SPS を学習に用いた各分類モデルは、2017–2018 年のデータに対して F1 スコアが 20 パーセントポイント以上低下し、2019–2020 年にかけて一定のスコアが回復する傾向が確認された。この傾向は、各組み合わせに共通しており、構文進化が検出性能に影響を及ぼしていることを示唆している。

6.3.2 誤分類分析によるクラスタ分布の結果

次に、F1 スコアの変動が最も顕著だった XGBoost + TF-IDF モデルについて、2017–2018 年における検出漏れ（FN）のクラスタ分布を集計した結果を表 2 に示す。全体の 67.2%が Cluster 1（難読・委任実行型）に属しており、次いで Cluster 0（状態チェック型）が 20.8%を占めていた。これらのクラスタは、2015–2016 年の学習時点ではほとんど出現していないか、構文的に大きく異なる特徴を持つため、モデルの汎化が困難であったと推察される。

以上の結果から、構文進化、特に 2017–2018 年に観察された異質なクラスタ（Cluster 1）の増加が、既存の検出器の検出性能を一時的に損なう要因となっていたことから、検出器の適用限界が定量的に示唆された。

7. 考察

7.1 SPS の設計傾向と回避行動の可能性

RQ1 の分析で明らかとなった構文傾向の時系列変化は、スマートポンジスキーム（SPS）の設計者が検出を回避する意図のもとで構文を変更している可能性を示唆する。特に、2017–2018 年に顕著に増加した Cluster 1（難読・委任実行型）や Cluster 3（偽装型）は、通常の検出器が学習していた構文パターンから逸脱しており、回避行動の表れと考えられる。

また、2019 年以降は Cluster 4（制御構文型）に収束しており、この変化には複数の解釈が可能である。一つは、難読構文に対して、検出器が適応できるようになったことで、悪意のある開発者がそれらの構文を避け、結果的に Cluster4 が相対的に増加し収束していったという可能性である。もう一つは逆に難読構文が依然として検出困難なままであるため、検出器が構文的に単純な SPS しか検出できておらず、検出漏れが生じている可能性である。前者は検出器の進展を示すが、後者であれば現行手法の限界を強く示唆するものであり、警戒が必要である。

7.2 PonziGuard の適応限界と構文的傾向の関係

本研究では、PonziGuard によって検出された SPS を対象に、構文的傾向の変化と検出性能の関係を分析し、その適応限界を明らかにした。

RQ2 では、2015–2016 年の SPS で学習した分類モデルを用いて検出性能を評価した結果、2017–2018 年には F1 スコアが大きく低下し、特に難読実行型（Cluster 1）に誤分類（False Negative）が集中していた。このことから、未学習の構文パターンが登場したことにより、汎化性能が著しく損なわれたと考えられる。

一方、2019–2020 年には F1 スコアが一定程度回復しており、これは再び制御構文型（Cluster 4）主流になり、学習データとの構文的一致度が高まり、分類器が再び高い性能を発揮できたためと考えられる。

いずれにせよ、検出器の性能が特定の構文パターンに依存しており、構文進化に対して脆弱であることは明らかである。特に、学習データに現れない構文群が出現した場合、モデルの汎化性能が低下し、見逃し（FN）が集中するリスクがある。

以上より、構文的に異質な SPS への対応には限界があり、構文の進化や多様化に追従できる検出器の設計が今後の課題となる。継続的なモデル更新や、構文進化への耐性強化は、スマートコントラクトの詐欺に対する検出手法の実効性と持続性を担保するうえで不可欠である。

7.3 本研究の限界

本研究にはいくつかの制約がある。第一に、分析対象は PonziGuard によって検出された SPS に限定されており、未検出のものは含まれていない。そのため、本研究で示した構文傾向や年次変化は、検出可能だった SPS の一部に基づくものであり、全体の傾向を反映してない可能性がある。また、PonziGuard の検出結果をラベルとしたため、全コントラクトが真に SPS だとは限らない。特にソースコードが非公開のものがあり、動作仕様や設計目的の妥当性を

直接確認することは難しい。

第二に、分析対象の SPS は、バイトコードが取得可能であった 742 件に限られており、取得できなかった 63 件は除外している。これらに特異な構文傾向が含まれる場合、本研究では反映されてない。ただし、除外対象の年代偏りは小さく、年次評価への影響は限定的と考えられる。

第三に、構文分析はオペコード列に基づくものであり、各クラスタの設計意図や詐欺性を直接示すものではない。特に難読化された Cluster 1 などはソースコードがすべて非公開であるなど限界がある。

第四に、コントラクトの展開時刻は Etherscan API を用いており、タイムスタンプには一定の誤差などの限界がある。特に再デプロイやプロキシの利用などがある場合、初回展開時刻に影響を与える可能性がある。ただし、先行研究でも詳細に明示されていないケースが多く、本研究が特に劣っているとは言い難い。

以上のように、本研究は検出済み SPS の構文傾向に基づく分析であり、構文的多様性や検出回避の網羅的把握には至っていない。今後は検出漏れの可能性を含む全体構造の分析や、ソースコードの限定的開示を前提とした詳細解析などが求められる。

7.4 今後の課題と展望

本研究では、PonziGuard の検出結果に基づき、スマートポンジスキーム（SPS）の構文進化と検出性能の関係を分析した。その結果、従来の高性能モデルでも構文の多様化により検出漏れが生じる可能性があることが示された。これを踏まえ、今後の検出器設計には以下の視点が求められる。

まず、2015–2016 年の学習モデルが 2017–2018 年の構文に対応できなかったことから、転移学習や継続学習などによる耐性・汎化性能の強化が必要である。また、特定クラスタにおける短期間の構文変化は、敵対的な実装の可能性を示唆しており、今後はこれに対応するため構文的特徴の選定や、多視点的な評価手法が重要となる。

さらに、より包括的な検出を行うためには、構文や年代に偏りのないラベル付きデータセットの整備が欠かせない。加えて、クラスタリングによる構文傾向の監視は、新たな詐欺傾向を早期に察知する補助的手段として有効であり、誤分類される新奇な構文を早期に抽出・警戒するためのモジュールとしての連携が期待される。

このように今後は、検出性能や早期検出だけではなく、構文進化への対応力を持つ堅牢な検出器の開発が、スマートポンジスキーム検出において重要となる。

8. まとめ

本研究では、最新のスマートポンジスキーム検出器である PonziGuard によって検出されたコントラクト群を対象に、その検出結果を構文的特徴の観点から分析することで、同手法の検出性能および適応限界を評価した。特に、年代ごとの構文進化と、ベースラインモデルによる補完的な分類実験を通じて、PonziGuard が構文的に複雑または新奇なパターンに対して検出漏れを起こす可能性があることを示唆した。また、構文的類似性と検出性能との関連性も確認され、検出器の汎化性能を確保するためには、構文の進化を踏まえた設計が不可欠であることが明らかとなった。これらの結果は、今後のスマートポンジスキーム検出技術において、構文的多様性への対応やモデルの継続的な更新などの重要性を示すものである。

謝辞　本研究は JSPS 科研費 JP25H01106 の助成を受けたものである.

参考文献

[1] Liang, R., Chen, J., He, K., Wu, Y., Deng, G., Du, R. and Wu, C.: Ponziguard: Detecting ponzi schemes on ethereum with contract runtime behavior graph (CRBG), Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, pp. 1–12 (2024).

[2] Bartoletti, M., Carta, S., Cimoli, T. and Saia, R.: Dissecting Ponzi schemes on Ethereum: identification, analysis, and impact, *Future Generation Computer Systems*, Vol. 102, pp. 259–277 (2020).

[3] Hu, T., Liu, X., Chen, T., Zhang, X., Huang, X., Niu, W., Lu, J., Zhou, K. and Liu, Y.: Transaction-based classification and detection approach for Ethereum smart contract, *Information Processing and Management*, Vol. 58, No. 2, pp. 1–19 (2021).

[4] Chen, W., Zheng, Z., Ngai, E. C.-H., Zheng, P. and Zhou, Y.: Exploiting blockchain data to detect smart ponzi schemes on ethereum, *IEEE Access*, Vol. 7, pp. 37575–37586 (2019).

[5] Wang, L., Cheng, H., Zheng, Z., Yang, A. and Zhu, X.: Ponzi scheme detection via oversampling-based long short-term memory for smart contracts, *Knowledge-Based Systems*, Vol. 228, pp. 1–12 (2021).

[6] Fan, S., Fu, S., Xu, H. and Cheng, X.: Al-SPSD: Anti-leakage smart Ponzi schemes detection in blockchain, *Information Processing and Management*, Vol. 58, No. 4, pp. 1–13 (2021).

[7] Zheng, Z., Chen, W., Zhong, Z., Chen, Z. and Lu, Y.: Securing the ethereum from smart ponzi schemes: Identification using static features, *ACM Transactions on Software Engineering and Methodology*, Vol. 32, No. 5, pp. 1–28 (2023).

[8] Fernández, A., Garcia, S., Herrera, F. and Chawla, N. V.: SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary, *Journal of artificial intelligence research*, Vol. 61, pp. 863–905 (2018).

[9] : XBlock: eXplore Blockchain Reliability — xblock.pro, `http://xblock.pro/#/dataset/PonziContractDataset`. Accessed 17-10-2024.

[10] Wang, L., Cheng, H., Sun, Z., Tian, A. and Yang, Z.: PSPL: A Ponzi scheme smart contracts detection approach via compressed sensing oversampling-based peephole LSTM, *Future Generation Computer Systems*, Vol. 166, pp. 1–12 (2025).

[11] PonziDetection: GitHub - PonziDetection/PonziGuard — github.com, https://github.com/PonziDetection/PonziGuard. Accessed 17-10-2024.

[12] etherscan.io: Ethereum (ETH) Blockchain Explorer — etherscan.io, https://etherscan.io/. Accessed 17-10-2024.

[13] Fisher, R. A.: On the interpretation of χ 2 from contingency tables, and the calculation of P, *Journal of the royal statistical society*, Vol. 85, No. 1, pp. 87–94 (1922).

[14] Benjamini, Y. and Hochberg, Y.: Controlling the false discovery rate: a practical and powerful approach to multiple testing, *Journal of the Royal statistical society: series B (Methodological)*, Vol. 57, No. 1, pp. 289–300 (1995).