Научная статья
УДК 004.42

МЕТОД ОБНАРУЖЕНИЯ ДУБЛИКАТОВ ИСХОДНОГО КОДА НА ОСНОВЕ АЛГОРИТМА СЛУЧАЙНОГО БЛУЖДАНИЯ

Яковлев Артем Владимирович;
Таров Евгений Викторович.
Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича, Санкт-Петербург, Россия.
✉**Израилов Константин Евгеньевич;**
Чечулин Андрей Алексеевич.
Санкт-Петербургский Федеральный исследовательский центр Российской академии наук, Санкт-Петербург, Россия
✉***konstantin.izrailov@mail.ru***

Аннотация. Статья посвящена задаче обнаружения экземпляров исходного кода, дублирующего заданный. В интересах этого рассматриваются существующие подходы поиска клонов кода, основанные на текстовом, лексическом, синтаксическом, метрическом и семантическом анализе. Основываясь на их критериальном сравнении, предлагается новый метод поиска дубликатов, в основу которого положен алгоритм случайного блуждания. Суть метода заключается в построении графов двух исходных кодов (где узлами являются лексические единицы текста, а ребрами – связи между ними), на которых затем применяется указанный алгоритм; дается описание метода в виде псевдокода. Проводится эксперимент по оценке работоспособности метода с использованием следующих метрик: Жаккара, разниц количества ребер, вершин и средней кластеризации графов, кратчайшего пути между их вершинами, а также сходства между графами. Сценарии экспериментов состоят из вычисления метрик для комбинаций экземпляров двух исходных кодов, их объединения и доли одного из них. Делаются выводы касательно применимости как самого метода, так и каждой из метрик оценки.

Ключевые слова: информационная безопасность, поиск дубликатов, случайное блуждание, исходный код

Для цитирования: Яковлев А.В., Таров Е.В., Израилов К.Е., Чечулин А.В. Метод обнаружения дубликатов исходного кода на основе алгоритма случайного блуждания // Науч.-аналит. журн. «Вестник С.-Петерб. ун-та ГПС МЧС России». 2023. № 2. С. 134–146.

Scientific article

METHOD FOR DETECTING SOURCE CODE DUPLICATES BASED ON THE RANDOM WALK ALGORITHM

Yakovlev Artem V.;
Tarov Evgeny V.
Saint-Petersburg state university of telecommunications them. prof. M.A. Bonch-Bruevich, Saint-Petersburg, Russia.
✉Izrailov Konstantin E.
Chechulin Andrey A.
Saint-Petersburg Federal research center of the Russian academy of sciences, Saint-Petersburg, Russia
✉konstantin.izrailov@mail.ru

Abstract. The article is devoted to the problem of source code finding copies that duplicates the given one. In the interests of this, the existing approaches for searching for code clones based on textual, lexical, syntactic, metric and semantic analysis are considered. Based on their criterion comparison, a new method for searching for duplicates is proposed, which is based on the random walk algorithm. The essence of the method is to build graphs of two source codes (where the nodes are the tokens of the text, and the edges are the links between them), on which the specified algorithm is then applied; the description of the method is given in the form of pseudocode. An experiment is being carried out to evaluate the performance of the method using the following metrics: Jaccard, differences in the number of edges, vertices and average clustering of graphs, the shortest path between their vertices, as well as similarities between graphs. Experimental scenarios consist of calculating metrics for combinations of two source codes instances, their union and the proportion of one of them. Conclusions are drawn regarding the applicability of both the method itself and each of the evaluation metrics.

Keywords: information security, duplicate search, random walk, source code

For citation: Yakovlev A.V., Tarov E.V., Izrailov K.E., Chechulin A.A. Method for detecting source code duplicates based on the random walk algorithm // Scientific and analytical journal «Vestnik Saint-Petersburg university of State fire service of EMERCOM of Russia». 2023. № 2. P. 134–146.

Введение

В настоящее время программное обеспечение (ПО) является неотъемлемой частью современной жизни. Оно используется практически везде – начиная от мобильных приложений и заканчивая крупными корпоративными системами [1]. Однако, наряду со всеми преимуществами, которые ПО может принести, существует и некоторый вид угроз [2], вытекающих из следующего противоречия. С одной стороны, повторное использование исходного кода программ в ряде случаев должно строго контролироваться; например, для препятствования копирования ошибок из предыдущих версий ПО [3] (что стало причиной известнейшей аварии европейской ракеты Ariane 5) или же для недопущения нарушения интеллектуальной собственности [4] и кражи чужого кода [5]. С другой стороны, современные технологии позволяют без особых усилий создать практически бесконечное число копий исходного кода ПО. Одним из способов противодействия этим угрозам может быть развитие методов поиска дубликатов исходного кода. Это позволит, в том числе, снизить общее число ошибок в коде, уменьшить его размер и увеличить структуризацию, ускорить работу; и как итог, повысится общая эффективность процесса разработки программных продуктов [6].

Исходя из вышесказанного, целью данного исследования является снижение степени незаконного дублирования исходного кода при разработке, распространении и эксплуатации ПО. Для достижения цели первым шагом выступает решение задачи по разработке метода обнаружения дубликатов исходного кода. Такой метод, в основу которого лег алгоритм случайного блуждания, и будет описан далее.

Современные методы поиска дубликатов

В настоящее время существует множество методов сравнения исходного кода, которые могут быть использованы для поиска дубликатов ПО [7–12]. В качестве основных методов можно выделить следующие:

– текстовый, использующий сравнение хеш-значений строк исходного кода [13];

– лексический, работающий уже с токенами исходного кода [14];

– синтаксический, оперирующий абстрактным синтаксическом деревом [15, 16];

– метрический, вычисляющий численные метрики для дерева абстрактного синтаксиса и ряда других графов;

– семантический, использующий графы зависимостей между операторами для выявления близких подграфов [17].

Несмотря на достаточное разнообразие подходов, применяемых в стандартных методах, ни один из них не дает полную гарантию в обнаружении дубликатов. Также исходного кода были сравнены по следующим интуитивно понятным критериям: точность и полнота определения, скорость работы, ресурсоёмкость и простора реализации. Оценка соответствия критериям проводилась по трехбалльной шкале, а интегральная оценка (как сумма оценок метода по всем критериям) показала, что «наилучшим» методом является текстовый, за которым идут лексический и синтаксический. Таким образом, целесообразно разработать новый метод, построенный на принципах их работы (по возможности, учитывающий достоинства и нивелирующий недостатки рассмотренных).

Новый метод

Предложим даже новый, потенциально перспективный метод поиска клонов исходных кодов на базе алгоритма случайного блуждания по графу [18–23]. Суть алгоритма заключается в том, чтобы рассмотреть коды как ориентированные графы, где вершины представляют собой токены, а ребра – связи между ними. При этом если в коде программы несколько одинаковых объектов, то новые узлы не строятся, а их соседи добавляются к уже существующему. Так, например, для следующего исходного кода: x=y+z+1, будет построен граф, состоящий из шести узлов, поскольку для оператора «+» будет создан лишь один узел. Графическое представление такого графа показано на рисунке.

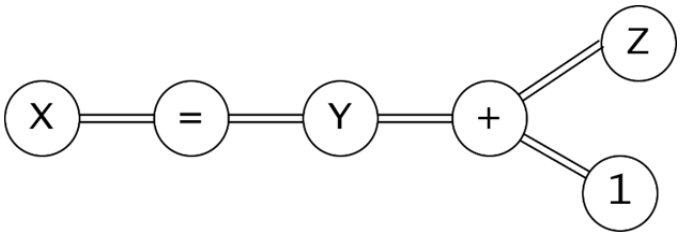

Рис. **Графическое представление графа для примера исходного кода**

После построения графа потребуется выбрать случайный стартовый узел в каждом графе и запустить для них алгоритм случайного блуждания по узлам. В процессе этого будет выбираться случайное ребро из текущей вершины и многократно производиться переход к следующей вершине по этому ребру. В результате для каждого графа будет получена последовательность вершин, которые были посещены в процессе случайного блуждания. Шаги данного метода на базе алгоритма случайного блуждания можно описать следующим образом.

Шаг 1. Подать на вход два исходного кода.

Шаг 2. Для каждого исходного кода создать ориентированный граф, где вершины – это токены, а ребра – связи между ними.

Шаг 3. В каждом графе исходных кодов выбрать случайный стартовый узел.

Шаг 4. Начать процесс случайного блуждания на каждом графе: из текущей вершины выбрать случайное ребро и перейти к следующей вершине.

Шаг 5. Задать число повторений (итераций) шагов 3, 4.

Шаг 6. Для каждого графа получить последовательность вершин, которые были посещены в процессе работы алгоритма.

Шаг 7. Вычислить метрики различия графов.

Шаг 8. Проанализировать метрики и сделать вывод о сходстве двух кодов.

Метрики сравнения

Для возможности оценки сходства двух кодов с точки зрения их графового представления необходимо использовать соответствующие метрики.

В качестве первой метрики выбран коэффициент Жаккара [24–26], который вычисляется, как отношение мощности пересечения множеств токенов в двух фрагментах кода к их объединению в соответствии со следующей формулой:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|},$$

где A и B – множества токенов в каждом из фрагментов кода.

Коэффициент Жаккара принимает значения от 0 до 1, где 0 означает полное отсутствие сходства, а 1 – полное совпадение. Таким образом, чем ближе значение коэффициента Жаккара к 1, тем более схожими являются два фрагмента кода. Обозначим метрику, как JC (*от англ.* Jaccard Coefficient).

Второй метрикой будет выступать разница между количеством ребер в двух графах. Так, если значение метрики близко к 0, то это означает, что графы содержат примерно одинаковое количество ребер. Обозначим данный коэффициент следующим образом: NED (*от англ.* Number of Edges Difference).

По аналогии со второй метрикой, третьей будет разница между количеством вершин в графах. Так, если значение метрики близко к 0, то графы содержат примерно одинаковое количество вершин. Обозначим метрику как NND (*от англ.* Number of Nodes Difference).

Четвертая – разница между средним коэффициентом кластеризации графов, который характеризует степень связности (плотности) их вершин. Так, если значение метрики близко к нулю, то графы имеют примерно одинаковый уровень связности. Обозначим метрику как CCD (*от англ.* Clustering Coefficient Difference).

Пятой метрикой будет разница между средним кратчайшим путем относительно всех пар вершин графов. Так, если значение этой метрики близко к нулю, то графы имеют примерно одинаковую структуру кратчайших путей. Обозначим метрику как ASPD (*от англ.* Average Shortest Path Difference).

И, наконец, шестой метрикой будем считать показатель сходства между графами [27], который определяется как отношение количества пересечений между случайными парами вершин в графах и общего количества шагов в случайном блуждании. Так, если значение этой метрики близко к нулю, то структуры графов сильно отличаются. Обозначим метрику как SG (*от англ.* Similarity of Graphs).

С учетом введенных метрик программная реализация метода может быть описана с помощью следующего псевдокода:

```
Input:
    Code1 – листинг первого исходного кода
    Code2 – листинг второго исходного кода
    num_steps – число итераций
Output:
    JS – метрика Жаккара
    NED – метрика в виде разницы количества ребер графов ИК
    NND – метрика в виде разницы количества вершин графов ИК
    CCD – метрика в виде разницы средней кластеризации графов ИК
    ASPD – метрика в виде разницы среднего кратчайшего пути между вершинами графов ИК
    SG – метрика в виде сходства между графами
Begin
 1:    graph1 = create_graph (Code1);
 2:    graph2 = create_graph (Code2);
 3:    current1 = get_random_node (graph1);
 4:    current2 = get_random_node (graph2);
 5:    intersections = 0;
 6:    For i = 1 To num_steps Do
 7:        neighbors1 = get_neighbors (graph1, current1);
 8:        neighbors2 = get_neighbors (graph2, current2);
 9:        If len(neighbors1) > 0 Then
10:            current1 = choose_random_node (neighbors1);
11:        EndIf
12:        If len(neighbors2) > 0 Then
13:            current2 = choose_random_node (neighbors2);
14:        EndIf
15:        If current1 == current2 Then
16:            intersections = intersections + 1;
17:        EndIf
18:    EndFor
19:    JC = calculate_jaccard_coefficient (graph1, graph2);
20:    NED = calculate_num_edges_diff (graph1, graph2);
21:    NND = calculate_num_nodes_diff (graph1, graph2);
22:    CCD = calculate_clustering_coeff_diff (graph1, graph2);
23:    ASPD = calculate_avg_shortest_path_diff (graph1, graph2);
24:    SG = (intersections / num_steps) * 100;
25:    Return JC, NED, NND, CCD, ASPD, SG.
End
```

Входными параметрами псевдокода являются два листинга исходных кодов, которые должны быть оценены на предмет дублирования, а также число итераций алгоритма. Выходными параметрами псевдокода являются значения всех пяти приведенных ранее метрик.

Принцип работы псевдокода является интуитивно понятным, тем не менее дадим построчное описание алгоритма его работы.

В строках 1 и 2 производится построение графов входных исходных кодов (*graph1* и *graph2*).

В строках 3 и 4 выбирается случайный узел в каждом из графов (*current1* и *current2*).

В строке 5 количество пересечений между случайными парами вершин (*intersections*) инициализируется числом 0.

В строке 6 начинается блок выполнения цикла с заданным числом итераций (*num_steps*).

В строках 7 и 8 выбираются соседи для текущих узлов в обоих графах (*neighbors1* и *neighbors2*).

В строках 9–11 проверяется, что у текущего узла 1-го графа есть соседи, из которых, затем, случайным образом выбирается новый текущий узел.

Аналогично в строках 12–14 выбирается текущий узел для 2-го графа.

В строке 15–17 производится сравнение текущих узлов графов, если они совпадают, то увеличивается количество пересечений на 1.

В строке 18 заканчивается блок выполнения цикла, начатый в строке 6.

В строке 19 вычисляется метрика Жаккара (JC).

В строке 20 вычисляется метрика в виде разницы количества ребер графов ИК (NED).

В строке 21 вычисляется метрика в виде разницы количества вершин графов ИК (NND).

В строке 22 вычисляется метрика в виде разницы средней кластеризации графов ИК (CCD).

В строке 23 вычисляется метрика в виде среднего кратчайшего пути между вершинами графов (ASPD).

В строке 24 вычисляется метрика в виде сходства между графами (SG).

В строке 25 все вычисленные метрики возвращаются из программы.

Эксперимент

Для оценки работоспособности предложенного метода проведем эксперимент по сравнению двух экземпляров исходного кода всеми метриками.

Программная реализация алгоритма метода была осуществлена на языке программирования Python версии 3.10 с использованием библиотек networkx и random для работы с графами. В качестве токенов рассматривались все элементы кода, разделенные пробелами. Введем следующие обозначения, используемые далее в эксперименте: A – листинг первого кода; B – листинг второго кода; C – суммарный листинг кода A и B (то есть C=A+B); D – часть кода B в процентном соотношении (то есть $D \subset B$). При этом коды A и B были абсолютно различны и имели минимум одинаковых токенов.

В качестве первого эксперимента рассчитаем значения всех метрик для исходных кодов A, B и C, используя все логически значимые сочетания данных кодов. Результат эксперимента представлен в табл. 1.

Таблица 1

Метрики для комбинаций исходных кодов A, B и C

Код 1	Код 2	JC	NED	NND	CCD	ASPD	SG
A	A	1.000	0	0	0.000	0.00	2.50
C	C	1.000	0	0	0.000	0.00	2.21
B	B	1.000	0	0	0.000	0.00	2.57
B	C	0.508	168	110	0.020	0.38	1.75
A	C	0.581	160	94	0.036	1.57	1.78
A	B	0.089	8	16	0.056	1.95	1.37

В качестве второго эксперимента рассчитаем значения всех метрик для исходных кодов A, B и D. При этом код D получался путем выборки содержания кода B от его начала; доля содержания менялась от 10 % до 90 % (что будет указано в скобках после D).

Значение метрик для двух кодов D и B приведено в табл. 2.

Таблица 2

Метрики для исходных кодов D и B

Код 1	Код 2	JC	NED	NND	CCD	ASPD	SG
D (10 %)	B	0.175	142	94	0.148	0.69	1.82
D (20 %)	B	0.316	120	78	0.113	0.49	1.82
D (30 %)	B	0.438	101	64	0.084	0.33	1.86
D (40 %)	B	0.570	79	49	0.009	0.11	2.18
D (50 %)	B	0.719	56	32	0.003	0.24	2.18
D (60 %)	B	0.807	40	22	0.014	0.25	2.16
D (70 %)	B	0.895	21	12	0.014	0.29	2.19
D (80 %)	B	0.982	5	2	0.008	0.02	2.21
D (90 %)	B	1.000	1	0	0.008	0.01	2.21

Аналогично, значение метрик для двух кодов D и A приведено в табл. 3.

Таблица 3

Метрики для комбинаций исходных кодов D и A

Код 1	Код 2	JC	NED	NND	CCD	ASPD	SG
D (10 %)	A	0.041	150	110	0.092	1.25	1.22
D (20 %)	A	0.051	128	94	0.057	1.46	1.09
D (30 %)	A	0.052	109	80	0.028	1.61	1.15
D (40 %)	A	0.071	87	65	0.065	1.83	1.52
D (50 %)	A	0.081	64	48	0.059	1.71	1.46
D (60 %)	A	0.099	48	38	0.041	1.69	1.42
D (70 %)	A	0.094	29	28	0.041	1.65	1.47
D (80 %)	A	0.091	13	18	0.047	1.92	1.42
D (90 %)	A	0.089	9	16	0.045	1.94	1.37

В качестве третьего эксперимента рассчитаем значения всех метрик также для кодов A, B и D. При этом код D получался путем выборки содержания кода B случайным образом; доля содержания менялась от 10 % до 90 % (что будет указано в скобках после D).

Значение метрик для двух кодов D и B приведено в табл. 4.

Таблица 4

Расчет метрик третьего эксперимента для случая кода B

Код 1	Код 2	JC	NED	NND	CCD	ASPD	SG
D (10 %)	B	0.175	141	94	0.148	0.10	2.62
D (20 %)	B	0.281	120	82	0.028	0.68	2.28
D (30 %)	B	0.395	98	69	0.095	0.76	2.21
D (40 %)	B	0.429	96	65	0.121	0.44	2.09
D (50 %)	B	0.605	56	45	0.049	0.85	2.44
D (60 %)	B	0.789	28	24	0.145	0.23	1.79
D (70 %)	B	0.763	10	27	0.012	1.02	2.33
D (80 %)	B	0.885	8	13	0.024	0.92	2.01
D (90 %)	B	0.947	27	6	0.001	0.89	2.21

Аналогично, значение метрик для двух кодов D и A приведено в табл. 5.

Таблица 5

Расчет метрик третьего эксперимента для случая кода A

Код 1	Код 2	JC	NED	NND	CCD	ASPD	SG
D (10 %)	A	0.051	150	114	0.037	3.27	1.81
D (20 %)	A	0.071	126	95	0.025	1.61	1.33
D (30 %)	A	0.084	106	80	0.072	2.21	1.42
D (40 %)	A	0.096	82	66	0.047	2.18	1.35
D (50 %)	A	0.069	61	59	0.057	2.58	1.41
D (60 %)	A	0.095	40	52	0.003	3.13	1.35
D (70 %)	A	0.088	19	38	0.031	2.71	1.17
D (80 %)	A	0.091	3	32	0.086	2.87	1.25
D (90 %)	A	0.083	21	26	0.031	2.82	1.16

Результаты эксперимента

Анализ результатов эксперимента позволяет сделать следующие выводы.

Во-первых, метрика SG имеет хорошие результаты детектирования дубликатов исходного кода при помощи алгоритма предложенного метода. Из табл. 1 можно видеть, что при сравнении абсолютно одинаковых исходных кодов значение SG принимает наибольшее значение (2,50 для A, 2,57 для B и 2,21 для C), в то время как для разных исходных кодов A и B – наименьшее (1,37). Следовательно, чем меньше значение SG, тем меньше одинаковых элементов в анализируемых кодах. Подобная корреляция присутствует и в результатах из табл. 2–5.

Во-вторых, метрика JC также очень хорошо подходит для детектирования дубликатов в ПО. Чем меньше значение данного коэффициента, тем больше вероятность отсутствия каких-либо похожих частей в программах и наоборот. Например, из табл. 1–5 видно, что для разных исходных кодов значение JC не превосходило 0,1, а для случаев присутствия хотя бы некоторой части схожих токенов – не было менее 0,1. При этом для абсолютно идентичных кодов JC был равен 1. Таким образом, данную метрику можно использовать для предложенного метода, например, определив ее порог в 0,1.

В-третьих, метрики NED и NND показывают себя не так хорошо, как SG и JC. Так, например, для абсолютно схожих кодов из табл. 1 данные метрики принимают нулевые значения, хотя в табл. 4 для случая рандомизированного кода D этим коэффициентам соответствуют значения, отличные от нуля.

В-четвертых, метрика CCD также может использоваться для обнаружения схожих токенов. Из табл. 1–5 видно, что чем ближе значение к нулю, тем больше вероятность присутствия в анализируемых кодах дубликатов. Например, следуя табл. 1 для случаев схожих кодов данный коэффициент принимал значение 0, в то время как для абсолютно различных кодов – 0.056. Однако следует учесть, что для случая рандомизированного кода D (табл. 4) значение CCD было равно 0,148, хотя с увеличением процента оно все сильнее близилось к нулю.

И, в-пятых, метрика ASPD может стать удовлетворительным критерием для сравнения двух исходных кодов в рамках предложенного метода – из табл. 1–5 для нее можно выделить порог в 1,5. Если расчет ASPD двух кодов оказался меньше 1,5, то можно сделать вывод об их достаточной схожести элементов, в противном же случае рассматриваемые программы не содержат дубликатов кодов.

Заключение

В данной работе были рассмотрены различные подходы к поиску клонов кода, которые основываются на текстовом, лексическом, синтаксическом, метрическом и семантическом методах. Каждый из этих методов имеет свои преимущества и недостатки. Таким образом, в зависимости от типа задачи и программы он может быть более или менее эффективен.

В статье был предложен новый метод поиска дубликатов исходного кода, основанный на алгоритме случайного блуждания. Данный метод позволяет выявлять клоны кода, учитывая не только текстовые и синтаксические сходства, но и семантические. В частности, вместо того, чтобы опираться только на токены исходного кода, также используется семантическая близость между блоками кода.

Для оценки работоспособности предложенного метода использованы различные метрики, коэффициент Жаккара, количество различающихся узлов и ребер, различия в коэффициентах кластеризации и среднем расстоянии до других узлов в графе кода и степень сходства (на основе вершин графов исходного кода и глубины случайного блуждания по ним).

Дальнейшие исследования могут быть направлены на сравнение предложенного метода с другими существующими подходами к поиску клонов кода и оценку его эффективности в различных условиях и задачах. Кроме того, возможны дополнительные усовершенствования (например, для его применимости к машинному коду программ [28, 29]) и оптимизации алгоритма случайного блуждания (например, в части применения машинного обучения [30]) для более точного и быстрого поиска клонов кода в больших проектах.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-29-06099.

Список источников

1. Цифровые технологии и проблемы информационной безопасности: монография / Т.И. Абдуллин [и др.]. СПб.: СПГЭУ, 2021. 163 с.

2. Защита информации в компьютерных системах: монография / М.В. Буйневич [и др.]. СПб.: СПГЭУ, 2017. 163 с.

3. Израилов К.Е. Моделирование программы с уязвимостями с позиции эволюции ее представлений. Ч. 1. Схема жизненного цикла // Труды учебных заведений связи. 2023. Т. 9. № 1. С. 75–93. DOI: 10.31854/1813-324X-2023-9-1-75-93.

4. Сойников М.А. Взыскание ущерба, причиненного преступлением против интеллектуальной собственности: процессуальные аспекты // Lex Russica (Русский закон). 2019. № 12 (157). С. 80–86.

5. Слета В.Д. Поддержка повторного использования кода на основе онтологического подхода // Современные информационные технологии. 2010. № 11. С. 178–181.

6. Романов Н.Е., Израилов К.Е., Покусов В.В. Система поддержки интеллектуального программирования: машинное обучение feat. Быстрая разработка безопасных программ // Информатизация и связь. 2021. № 5. С. 7–17. DOI: 10.34219/2078-8320-2021-12-5-7-16.

7. Шуваев Ф.Л., Татарка М.В. Анализ математических моделей случайных графов, применяемых в имитационном моделировании информационно-коммуникационных сетей // Науч.-аналит. журн. «Вестник С.-Петерб. ун-та ГПС МЧС России». 2020. № 2. С. 67–77.

8. Suttichaya V., Eakvorachai N., Lurkraisit T. Source Code Plagiarism Detection Based on Abstract Syntax Tree Fingerprintings // The proceedings of 17th International Joint Symposium on Artificial Intelligence and Natural Language Processing (Chiang Mai, Thailand, 5–7 November 2022). 2022. P. 1–6. DOI: 10.1109/iSAI-NLP56921.2022.9960266.

9. Nishi M.A., Ciborowska A., Damevski K. Characterizing Duplicate Code Snippets between Stack Overflow and Tutorials // The proceedings of 16th International Conference

on Mining Software Repositories (Montreal, QC, Canada, 25–31 May 2019). 2019. P. 240–244. DOI: 10.1109/MSR.2019.00048.

10. Raheja K., Tekchandani R.K. An efficient code clone detection model on Java byte code using hybrid approach // The proceedings of Confluence 2013: The Next Generation Information Technology Summit (Noida, 26–27 September 2013). 2013. P. 16–21. DOI: 10.1049/cp.2013.2287.

11. Wang H., Zhong J., Zhang D. A Duplicate Code Checking Algorithm for the Programming Experiment // The proccedings of Second International Conference on Mathematics and Computers in Sciences and in Industry (Sliema, Malta, 17 August 2015). 2015. P. 39–42. DOI: 10.1109/MCSI.2015.12.

12. Moshkin V., Kalachev V., Zarubin A. Automation of Program Code Analysis Using Machine Learning Methods // The proocedings of International Russian Automation Conference (Sochi, Russian Federation, 4–10 September 2022). 2022. P. 404–408. DOI: 10.1109/RusAutoCon54946.2022.9896360.

13. Израилов К.Е., Гололобов Н.В., Краскин Г.А. Метод анализа вредоносного программного обеспечения на базе Fuzzy Hash // Информатизация и связь. 2019. № 2. С. 36–44.

14. Хилько В.О., Шаров И.А. Поиск дубликатов в исходных кодах программ // Региональная информатика и информационная безопасность. Вып. 4. 2017. С. 184–185.

15. Лисс А.Р., Андрианов И.А. Анализ и разработка методов поиска дубликатов в программном коде // Известия СПбГЭТУ ЛЭТИ. 2010. № 7. С. 55–61.

16. Вахрушев И.Н. Использование суффиксных деревьев для поиска дублирующихся фрагментов кода // Системы управления и информационные технологии. 2012. № 4 (50). С. 55–58.

17. Бородащенко А.Ю. Анализ текстов на семантическое сходство на основе аппарата теории графов // Известия Орловского государственного технического университета. Сер.: Информационные системы и технологии. 2008. № 1-2. С. 46–52.

18. De J., Zhang X., Lin F., Cheng L. Transduction on Directed Graphs via Absorbing Random Walks // IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 40. № 7. P. 1770–1784. DOI: 10.1109/TPAMI.2017.2730871.

19. Gori M., Maggini M., Sarti L. Exact and approximate graph matching using random walks // IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 27. № 7. P. 1100–1111. DOI: 10.1109/TPAMI.2005.138.

20. Lambiotte R., Delvenne J.-C., Barahona M. Random Walks, Markov Processes and the Multiscale Modular Organization of Complex Networks // IEEE Transactions on Network Science and Engineering. Vol. 1. № 2. P. 76–90. DOI: 10.1109/TNSE.2015.2391998.

21. Котов Э.М. Методы анализа гиперссылок при информационном поиске в глобальной сети // Известия ЮФУ. Технические науки. 2012. № 4 (129). С. 233–237.

22. Гасников А.В., Дмитриев Д.Ю. Об эффективных рандомизированных алгоритмах поиска вектора PageRank // Журнал вычислительной математики и математической физики. 2015. Т. 55. № 3. С. 355. DOI: 10.7868/S0044466915030060.

23. Кузьминова М.В. Периодические динамические графы. Задачи о случайных блужданиях и о кратчайших путях // Известия высших учебных заведений. Северо-Кавказский регион. Сер.: Естественные науки. 2008. № 2 (144). С. 16–21.

24. Немченко Д.С. Выявление общих черт пользователей для рекомендательной системы на примере коэффициента сходства Жаккара // Вестник магистратуры. 2023. № 3-1 (138). С. 14–15.

25. Кайда А.Ю. Анализ сходства текстов на основе коэффициента Жаккара-Танимото // Математические методы и модели техники, технологий и экономики: материалы Всерос. студ. науч.-практ. конф. 2022. С. 115–118.

26. Samanthula B.K., Jiang W. Secure Multiset Intersection Cardinality and its Application to Jaccard Coefficient // IEEE Transactions on Dependable and Secure Computing. Vol. 13. № 5. P. 591–604. DOI: 10.1109/TDSC.2015.2415482.

27. Самохин М.В. Вычисление сходства помеченных графов и их проекций // Научно-техническая информация. Сер. 2: Информационные процессы и системы. 2006. № 3. С. 1–12.

28. Kotenko I., Izrailov K., Buinevich M. The Method and Software Tool for Identification of the Machine Code Architecture in Cyberphysical Devices // Journal of Sensor and Actuator Networks. 2023. Vol. 12. Iss. 1. P. 11. DOI: 10.3390/jsan12010011.

29. Kotenko I., Izrailov K., Buinevich M. Analytical Modeling for Identification of the Machine Code Architecture of Cyberphysical Devices in Smart Homes // Sensors. 2022. Vol. 22. Iss. 3. P. 1017. DOI: 10.3390/s22031017.

30. Kotenko I., Izrailov K., Buinevich M. Static Analysis of Information Systems for IoT Cyber Security: A Survey of Machine Learning Approaches // Sensors. 2022. Vol. 22. Iss. 4. P. 1335. DOI: 10.3390/s22041335.

References

1. Cifrovye tekhnologii i problemy informacionnoj bezopasnosti: monografiya / T.I. Abdullin [i dr.]. SPb.: SPGEU, 2021. 163 s.

2. Zashchita informacii v komp'yuternyh sistemah: monografiya / M.V. Bujnevich [i dr.]. SPb.: SPGEU, 2017. 163 s.

3. Izrailov K.E. Modelirovanie programmy s uyazvimostyami s pozicii evolyucii ee predstavlenij. Ch. 1. Skhema zhiznennogo cikla // Trudy uchebnyh zavedenij svyazi. 2023. T. 9. № 1. S. 75–93. DOI: 10.31854/1813-324X-2023-9-1-75-93.

4. Sojnikov M.A. Vzyskanie ushcherba, prichinennogo prestupleniem protiv intellektual'noj sobstvennosti: processual'nye aspekty // Lex Russica (Russkij zakon). 2019. № 12 (157). S. 80–86.

5. Sleta V.D. Podderzhka povtornogo ispol'zovaniya koda na osnove ontologicheskogo podhoda // Sovremennye informacionnye tekhnologii. 2010. № 11. S. 178–181.

6. Romanov N.E., Izrailov K.E., Pokusov V.V. Sistema podderzhki intellektual'nogo programmirovaniya: mashinnoe obuchenie feat. Bystraya razrabotka bezopasnyh programm // Informatizaciya i svyaz'. 2021. № 5. S. 7–17. DOI: 10.34219/2078-8320-2021-12-5-7-16.

7. Shuvaev F.L., Tatarka M.V. Analiz matematicheskih modelej sluchajnyh grafov, primenyaemyh v imitacionnom modelirovanii informacionno-kommunikacionnyh setej // Nauch.-analit. zhurn. «Vestnik S.-Peterb. un-ta GPS MCHS Rossii». 2020. № 2. S. 67–77.

8. Suttichaya V., Eakvorachai N., Lurkraisit T. Source Code Plagiarism Detection Based on Abstract Syntax Tree Fingerprintings // The proceedings of 17th International Joint Symposium on Artificial Intelligence and Natural Language Processing (Chiang Mai, Thailand, 5–7 November 2022). 2022. P. 1–6. DOI: 10.1109/iSAI-NLP56921.2022.9960266.

9. Nishi M.A., Ciborowska A., Damevski K. Characterizing Duplicate Code Snippets between Stack Overflow and Tutorials // The proceedings of 16th International Conference on Mining Software Repositories (Montreal, QC, Canada, 25-31 May 2019). 2019. P. 240–244. DOI: 10.1109/MSR.2019.00048.

10. Raheja K., Tekchandani R.K. An efficient code clone detection model on Java byte code using hybrid approach // The proceedings of Confluence 2013: The Next Generation Information Technology Summit (Noida, 26-27 September 2013). 2013. P. 16–21. DOI: 10.1049/cp.2013.2287.

11. Wang H., Zhong J., Zhang D. A Duplicate Code Checking Algorithm for the Programming Experiment // The proccedings of Second International Conference on Mathematics and Computers in Sciences and in Industry (Sliema, Malta, 17 August 2015). 2015. P. 39–42. DOI: 10.1109/MCSI.2015.12.

12. Moshkin V., Kalachev V., Zarubin A. Automation of Program Code Analysis Using Machine Learning Methods // The proocedings of International Russian Automation Conference

(Sochi, Russian Federation, 4–10 September 2022). 2022. P. 404–408. DOI: 10.1109/RusAutoCon54946.2022.9896360.

13. Izrailov K.E., Gololobov N.V., Kraskin G.A. Metod analiza vredonosnogo programmnogo obespecheniya na baze Fuzzy Hash // Informatizaciya i svyaz'. 2019. № 2. S. 36–44.

14. Hil'ko V.O., Sharov I.A. Poisk dublikatov v iskhodnyh kodah programm // Regional'naya informatika i informacionnaya bezopasnost'. Vyp. 4. 2017. S. 184–185.

15. Liss A.R., Andrianov I.A. Analiz i razrabotka metodov poiska dublikatov v programmnom kode // Izvestiya SPbGETU LETI. 2010. № 7. S. 55–61.

16. Vahrushev I.N. Ispol'zovanie suffiksnyh derev'ev dlya poiska dubliruyushchihsya fragmentov koda // Sistemy upravleniya i informacionnye tekhnologii. 2012. № 4 (50). S. 55–58.

17. Borodashchenko A.Yu. Analiz tekstov na semanticheskoe skhodstvo na osnove apparata teorii grafov // Izvestiya Orlovskogo gosudarstvennogo tekhnicheskogo universiteta. Ser.: Informacionnye sistemy i tekhnologii. 2008. № 1-2. S. 46–52.

18. De J., Zhang X., Lin F., Cheng L. Transduction on Directed Graphs via Absorbing Random Walks // IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 40. № 7. P. 1770–1784. DOI: 10.1109/TPAMI.2017.2730871.

19. Gori M., Maggini M., Sarti L. Exact and approximate graph matching using random walks // IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 27. № 7. P. 1100–1111. DOI: 10.1109/TPAMI.2005.138.

20. Lambiotte R., Delvenne J.-C., Barahona M. Random Walks, Markov Processes and the Multiscale Modular Organization of Complex Networks // IEEE Transactions on Network Science and Engineering. Vol. 1. № 2. P. 76–90. DOI: 10.1109/TNSE.2015.2391998.

21. Kotov E.M. Metody analiza giperssylok pri informacionnom poiske v global'noj seti // Izvestiya YUFU. Tekhnicheskie nauki. 2012. № 4 (129). S. 233–237.

22. Gasnikov A.V., Dmitriev D.Yu. Ob effektivnyh randomizirovannyh algoritmah poiska vektora PageRank // Zhurnal vychislitel'noj matematiki i matematicheskoj fiziki. 2015. T. 55. № 3. S. 355. DOI: 10.7868/S0044466915030060.

23. Kuz'minova M.V. Periodicheskie dinamicheskie grafy. Zadachi o sluchajnyh bluzhdaniyah i o kratchajshih putyah // Izvestiya vysshih uchebnyh zavedenij. Severo-Kavkazskij region. Ser.: Estestvennye nauki. 2008. № 2 (144). S. 16–21.

24. Nemchenko D.S. Vyyavlenie obshchih chert pol'zovatelej dlya rekomendatel'noj sistemy na primere koefficienta skhodstva Zhakkara // Vestnik magistratury. 2023. № 3-1 (138). S. 14–15.

25. Kajda A.Yu. Analiz skhodstva tekstov na osnove koefficienta Zhakkara-Tanimoto // Matematicheskie metody i modeli tekhniki, tekhnologij i ekonomiki: materialy Vseros. stud. nauch.-prakt. konf. 2022. S. 115–118.

26. Samanthula B.K., Jiang W. Secure Multiset Intersection Cardinality and its Application to Jaccard Coefficient // IEEE Transactions on Dependable and Secure Computing. Vol. 13. № 5. P. 591–604. DOI: 10.1109/TDSC.2015.2415482.

27. Samohin M.V. Vychislenie skhodstva pomechennyh grafov i ih proekcij // Nauchno-tekhnicheskaya informaciya. Ser. 2: Informacionnye processy i sistemy. 2006. № 3. S. 1–12.

28. Kotenko I., Izrailov K., Buinevich M. The Method and Software Tool for Identification of the Machine Code Architecture in Cyberphysical Devices // Journal of Sensor and Actuator Networks. 2023. Vol. 12. Iss. 1. P. 11. DOI: 10.3390/jsan12010011.

29. Kotenko I., Izrailov K., Buinevich M. Analytical Modeling for Identification of the Machine Code Architecture of Cyberphysical Devices in Smart Homes // Sensors. 2022. Vol. 22. Iss. 3. P. 1017. DOI: 10.3390/s22031017.

30. Kotenko I., Izrailov K., Buinevich M. Static Analysis of Information Systems for IoT Cyber Security: A Survey of Machine Learning Approaches // Sensors. 2022. Vol. 22. Iss. 4. P. 1335. DOI: 10.3390/s22041335.

Информация о статье:
Статья поступила в редакцию: 21.04.2023; одобрена после рецензирования: 21.05.2023;
принята к публикации: 22.05.2023

The information about article:
The article was submitted to the editorial office: 21.04.2023; approved after review: 21.05.2023;
accepted for publication: 22.05.2023

Информация об авторах:
Яковлев Артем Владимирович, студент Санкт-Петербургского государственного университета телекоммуникаций им. проф. М.А. Бонч-Бруевича (193232, Санкт-Петербург, пр. Большевиков, д. 22/1), e-mail: 89062703467@mail.ru, https://orcid.org/0009-0006-5590-5095
Таров Евгений Викторович, студент Санкт-Петербургского государственного университета телекоммуникаций им. проф. М.А. Бонч-Бруевича (193232, Санкт-Петербург, пр. Большевиков, д. 22/1), e-mail: tarov25@mail.ru, https://orcid.org/0009-0003-9631-3609
Израилов Константин Евгеньевич, старший научный сотрудник лаборатории проблем компьютерной безопасности Санкт-Петербургского Федерального исследовательского центра Российской академии наук (199178, Санкт-Петербург, В.О., 14-я линия, 39), кандидат технических наук, доцент, e-mail: konstantin.izrailov@mail.ru, https://orcid.org/0000-0002-9412-5693
Чечулин Андрей Алексеевич, ведущий научный сотрудник лаборатории проблем компьютерной безопасности Санкт-Петербургского Федерального исследовательского центра Российской академии наук (199178, Санкт-Петербург, В.О., 14-я линия, 39), кандидат технических наук, доцент, e-mail: chechulin@comsec.spb.ru, https://orcid.org/0000-0001-7056-6972

Information about authors:
Yakovlev Artem V., student of the Saint-Petersburg state university of telecommunications them. prof. M.A. Bonch-Bruevich, Saint-Petersburg, Russia (193232, Saint-Petersburg, pr. Bolshevikov, 22/1), e-mail: 89062703467@mail.ru, https://orcid.org/0009-0006-5590-5095
Tarov Evgeny V., student of the Saint-Petersburg state university of telecommunications them. prof. M.A. Bonch-Bruevich, Saint-Petersburg, Russia (193232, Saint-Petersburg, pr. Bolshevikov, 22/1), e-mail: tarov25@mail.ru, https://orcid.org/0009-0003-9631-3609
Izrailov Konstantin E., senior researcher at the computer security problems laboratory of Saint-Petersburg Federal research center of the Russian academy of sciences (199178, Saint-Petersburg, 14-th linia 39), candidate of technical sciences, docent, e-mail: konstantin.izrailov@mail.ru, https://orcid.org/0000-0002-9412-5693
Chechulin Andrey A., leading researcher at the computer security problems laboratory of Saint-Petersburg Federal research center of the Russian academy of sciences (199178, Saint-Petersburg, 14-th linia, 39), candidate of technical sciences, docent, e-mail: chechulin@comsec.spb.ru, https://orcid.org/0000-0001-7056-6972