ИТОГИ НАУКИ И ТЕХНИКИ.
Современная математика и ее приложения.
Тематические обзоры.
Том 232 (2024). С. 140–152
DOI: 10.36535/2782-4438-2024-232-140-152

УДК 004.85

МЕТОД ГЛУБОКОГО ОБУЧЕНИЯ ДЛЯ ВЫЯВЛЕНИЯ АНОМАЛИЙ В ФУНКЦИОНИРОВАНИИ КОМПЬЮТЕРНЫХ СИСТЕМ

© 2024 г. **О. Е. ГОРОХОВ, М. И. ПЕТРОВСКИЙ, И. В. МАШЕЧКИН**

Аннотация. Задача обнаружения аномального поведения крупных программных систем может быть сведена к задаче обнаружения аномалий в потоках текстовых данных. В работе предлагается подход, основанный на комбинации глубокого обучения (автокодировщика с использованием сверточных нейронных сетей и однослойного полносвязного декодировщика) и подходов, основанных на нечетком методе кластеризации. Предложенное решение позволяет эффективно строить векторные представления групп последовательных событий и определять выбросы в данных за счет разработанного слоя, основанного на методах нечеткой кластеризации и радиально-базисных функций.

Ключевые слова: обнаружение аномалий, анализ системных журналов, глубокое обучение, нейронные сети.

DEEP LEARNING METHOD FOR IDENTIFYING ANOMALIES IN OPERATING COMPUTER SYSTEMS

© 2024 **O. E. GOROKHOV, M. I. PETROVSKII, I. V. MASHECHKIN**

Abstract. The problem of detecting anomalous behavior in large software systems can be reduced to the problem of detecting anomalies in text data streams. In this paper, we propose an approach based on a combination of deep learning (an autoencoder using convolutional neural networks and a single-layer fully connected decoder) and approaches based on the fuzzy clustering method. The solution proposed allows one to construct vector representations of groups of sequential events and identify outliers in the data using a developed layer based on fuzzy clustering and radial basis functions methods.

Keywords and phrases: anomaly detection, system log analysis, deep learning, neural networks.

AMS Subject Classification: 68T07

1. Введение. В последнее время возрастает значение различных компьютерных систем в деятельности человека. Они позволяют организовать оперативное общение людей, автоматизировать технологические процессы, обеспечить поддержку образования, медицины и других важных сфер жизни общества.

Особую роль играют программные системы, которые обеспечивают хранение и обработку больших данных, взаимодействуют с большим количеством пользователей. Возникновение ошибок в них может повлиять на многих пользователей и привести к значительным финансовым потерям. Поэтому, наиболее актуальной является проблема поддержания надежности и безопасности подобных систем.

Как правило, обнаружение ошибок в компьютерных системах возможно путем поиска аномального поведения на основании анализа происходящих событий, информация о которых хранится в сложно структурированном виде в специальных журналах. Эта информация может быть представима в текстовом виде. Таким образом, задача поиска аномального поведения системы может

быть сведена к задаче обнаружения аномалий в потоке текстовых описаний происходящих событий. в При этом важно отметить, что объем журналов может быть очень большим. Поэтому ручной поиск аномалий не представляется возможным [6,12,14]. Следовательно, нужно разработать алгоритм, позволяющий автоматически определять подобное аномальное поведение систем. Анализ существующих работ показывает, что в этой области эффективными являются подходы, основанные на глубоком обучении. Именно им и посвящена данная работа.

2. Анализ существующих работ. На основании анализа существующих работ можно утверждать, что общая схема обнаружения аномального поведения систем состоит из следующих этапов [1, 12]:

(i) построение модели данных;
(ii) построение модели нормального поведения системы;
(iii) обучение модели обнаружения аномального поведения;
(iv) применение модели обнаружения аномального поведения.

На этапе построения модели данных важно учитывать природу входных данных, которые, как было отмечено выше, представляют собой текстовые описания происходящих в системе событий. Следовательно, на этом этапе могут применяться классические методы анализа текстовых данных (токенизация, лемматизация; см. [7, 18]).

Кроме того, важно помнить, что входные текстовые данные описывают события и обладают рядом специфических особенностей [7]:

(i) описания событий сильно зависят от конкретной системы, поэтому для разработки универсального алгоритма необходимо минимизировать множество алгоритмов предобработки данных;
(ii) события часто похожи друг на друга и отличаются только наличием специальных идентификаторов связанных объектов (процессов, подсистем и т. д.),
(iii) множество уникальных событий ограничено относительно небольшим размером словаря (не больше 1000);
(iv) входные данные представляют собой поток описаний связанных между собой событий.

Указанные особенности необходимо также учитывать на этапе построения модели в данных. В частности, в качестве предобработки можно использовать только удаление различных идентификаторов с помощью регулярных выражений. Поскольку уникальных событий в системе, как правило, не очень много, в качестве токена можно использовать отдельно взятые события.

После выполнения токенизации необходимо провести векторизации входных данных. На этом этапе применимы классические методы векторизации текста (One-hot кодирование отдельных событий, модели описаний событий как текстовые модели множества либо последовательности слов) [2].

Следующий этап заключается в построении признакового описания последовательности взаимосвязанных событий. Полученное информативное векторное представление указанных последовательностей и может рассматриваться в качестве модели данных. На этом этапе широко применяются подходы, основанные на сокращении размерности признакового пространства (латентное разложение Дирихле, матричные разложения и др.) [2]. Также в последнее время применяются подходы, основанные на глубоком обучении (сверточные нейронные сети, архитектуры трансформеров с использованием механизма самовнимания) [19].

После построения модели данных необходимо построить и обучить модель описания нормального поведения системы. Для этого, необходимо корректно сформулировать критерий аномального поведения системы. Здесь важно отметить, что в большинстве практических задач очень трудно заранее корректно описать аномальное поведение систем. Однако, можно подобрать данные, в которых поведение системы было нормальным на протяжении практически всего времени работы системы.

Поэтому на данном этапе следует использовать классический подход сведения поставленной задачи к задаче одноклассовой классификации [1]. Особый интерес представляет классический одноклассовый метод опорных векторов [1], который может рассматриваться в качестве базового

решения. Также стоит обратить внимание на методы нечеткой кластеризации, которые развиваются авторами данной работы [8,17]. Также в последнее время наиболее эффективными являются различные модификации архитектур трансформеров [5]. Именно эти подходы и будут рассмотрены далее.

Одноклассовый метод опорных векторов [13] представляет собой некоторую модификацию классического метода опорных векторов для задач одноклассовой классификации и позволяет строить разделяющую гиперплоскость на основании решения следующей задачи оптимизации:

$$\min \frac{1}{2} * ||w||^2 + \frac{1}{vl} \sum_{i=1}^{l} \xi_i - \rho, \quad (w * \Phi(x_i)) \geqslant \rho - \xi_i; \quad i = 1, 2, \ldots, l; \quad \xi_i \geqslant 0.$$

Здесь $x_i \in \mathbb{R}$ — объекты обучающей выборки, $v \in (0, 1)$ — процент исключений в обучающей выборке, l — количество объектов в обучающей выборке, ξ_i — некоторые штрафующие переменные, $\Phi : \mathbb{R}^N \to H$ — некоторая ядровая функция, осуществляющая отображение первоначального пространства признаков в пространство большей размерности H. Если w и ρ — решение поставленной задачи, то решающая функция выглядит следующим образом:

$$f(x) = \text{sign}(w * \Phi(x) - \rho).$$

Методы нечеткой кластеризации [3,17] позволяют строить так называемые нечеткие кластеры и определять степень аномальности входящих в обучающую выборку событий. При этом, лучший результат можно получить с помощью использования эллиптических кластеров, учитывающих матрицу ковариаций данных обучающей выборки [8].

Суть метода заключается в следующем. Рассмотрим конечную выборку входных данных: $\{x_i \in X \mid i = 1, \ldots, N\}$. Здесь N — размер выборки, X — множество значений из пространства признаков. Необходимо построить единый кластер эллипсоидальной формы, включающий все исследуемые образцы таким образом, что степень типичности u_i каждого образца x_i вычисляется путем решения следующей задачи оптимизации:

$$\min_{U,a,\mu} E(u, a, \mu) = \mu \sum_{i=1}^{N} u_i^m * ||x_i - a||_C^2.$$

Здесь u_i^m — степень типичности образца x_i, $m > 1$ — степень нечеткости, U — множество степеней типичности образцов, $||x_i - a||_C^2$ — квадрат расстояния Махалонобиса от вектора x_i до центра кластера a, вычисляемый по формуле

$$D_i(a) = ||x_i - a||_C^2 = (x_i - a)^T * C^{-1} * (x_i - a),$$

где C — матрица ковариаций.

В ходе исследований, проведенных в изученной литературе [8], степень типичности объекта может определяться по следующей формуле:

$$u_i = \left[1 + \left(\frac{D_i(a)}{\mu} \right)^{1/(m-1)} \right]^{-1},$$

где μ — радиус кластера, на котором образы анализируемых объектов имеют степень типичности, равную 0,5. Этот радиус вычисляется на основании расстояний каждого объекта обучающей выборки до найденного центра распределения.

Подходы, основанные на архитектуре трансформеров с механизмом самовнимания (см. [5,19]), позволяют строить информативные векторные представления потока событий, учитывающие взаимные связи событий. Данный алгоритм обучается на задаче маскированного моделирования, заключающейся в предсказании пропущенных событий в рассматриваемой последовательности. При этом в последнее время наиболее эффективными являются подходы, основанные на так называемом многоголовом самовнимании. Такие архитектуры представляют собой параллельно связанные «головы» (элементы, вычисляющие степень внимания). Каждая «голова» позволяет

определять степень внимания, исходя из различных особенностей данных. Формально каждая «голова» представляет собой скалярное произведение:

$$\texttt{head}_l = \texttt{Attention}(X^j W_l^Q, X^j W_l^K, X^j W_l^V),$$
$$\texttt{Attention}(Q, K, V) = \texttt{softmax}\left(\frac{QK^T}{\sqrt{d_v}}\right)V.$$

Здесь $X^j \in \mathbb{R}^{T*d}$ — информативное представление последовательности событий; W_l^Q, W_l^K и W_l^V — матрицы весов для линейной проекции размерности $\mathbb{R}^{d*d_v}$ для «головы» с номером l; d_v — размерность одной «головы».

Механизм многоголового самовнимания производит объединение всех голов линейное отображение в некоторое новое пространство размерности d_O с помощью матрицы весов $W^O \in \mathbb{R}^{hd_v*d_O}$:

$$f(X^j) = \texttt{Concat}(\texttt{head}_1, \ldots, \texttt{head}_H)W^O.$$

Далее, для предсказания пропущенного события используется полносвязный слой с функцией активации ReLU:

$$\texttt{transformer\_layer}(X^j) = \texttt{ReLU}(f(X^j)W_1)W_2.$$

Здесь W_1 и W_2 — некоторые матрицы весов.

Классическая архитектура трансформера состоит из нескольких таких слоев transformer_layer и позволяет строить информативное векторное представление для события x_t^j по следующему правилу:

$$h_j^t = \texttt{Transformer}(x_t^j)$$

Как было сказано выше, в процессе обучения архитектуры трансформера для поставленной проблемы решаются две задачи: маскированное моделирование пропущенных событий и построение гиперсферы для определения кластера нормальных событий. Таким образом, целевая функция выглядит следующим образом:

$$\mathcal{L} = \mathcal{L}_{\texttt{MLKP}} + \alpha * \mathcal{L}_{\texttt{VHM}}.$$

Здесь α — гиперпараметр; $\mathcal{L}_{\texttt{MLKP}}$ — оценка предсказания векторного представления пропущенного события с вектором $h^j_{[\texttt{MASK}_i]}$:

$$\mathcal{L}_{\texttt{MLKP}} = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{M} y^j_{[\texttt{MASK}_i]} \log \hat{y}^j_{[\texttt{MASK}_i]}, \quad \hat{y}^j_{[\texttt{MASK}_i]} = \texttt{Softmax}(W_C h^j_{[\texttt{MASK}_i]} + b_C).$$

Здесь $\hat{y}^j_{[\texttt{MASK}_i]}$ — вероятность предсказанного события; W_C и b_C — обучаемые параметры; $y^j_{[\texttt{MASK}_i]}$ — векторное представление реального замаскированного события с номером i; M — общее количество замаскированных событий в последовательности с номером j; $\mathcal{L}_{\texttt{VHM}}$ — функция для построения кластера нормальных событий:

$$\mathcal{L}_{\texttt{VHM}} = \frac{1}{N}\sum_{j=1}^{N} ||h^j_{\texttt{DIST}} - c||^2.$$

Здесь $h^j_{\texttt{DIST}}$ — информативное векторное представление всей информации о событиях в последовательности с номером j; $c = \texttt{Mean}(h^j_{\texttt{DIST}})$ — центр распределения нормальных событий.

Однако большинство рассмотренных подходов работают лишь в режиме бинарной классификации, когда на этапе обучения известны примеры аномального поведения. Они не приспособлены к задаче одноклассовой классификации, и показывают в ней не очень хороший результат. Кроме того, подходы, основанные на архитектуре трансформера достатоно затратны по времени обучения и применения, что вызывает проблемы при реальном применении указанных подходов. В данной работе предлагается рассмотреть подход, учитывающий указанные недостатки и позволяющий эффективно решать задачу одноклассовой классификации.

3. Предлагаемое решение. Как было сказано выше, методы глубокого обучения широко применяются для анализа сложно структурированных данных. При этом, для задач обнаружения аномалий в данных системных журналов используются описанные выше методы, основанные на архитектуре трансформера с использованием механизма самовнимания. Однако, данный механизм требует регулярного вычисления значения внимания для каждого отдельного блока и сравнения его с остальными блоками на каждой итерации обучения, что является достаточно сложным с точки зрения вычислительной сложности.

В настоящей работе предлагается альтернативный подход, который напоминает указанные архитектуры. Однако, в качестве алгоритма построения модели данных предлагается использовать операцию параллельной свертки, предложенной в работе Ю. Ким и широко применяемую в анализе текстовых данных [9]. Данная операция также позволяет учесть порядок событий, что важно в рамках поставленной задачи. Кроме того, признаки, получаемые в процессе обучения каждой сверточной части автоматически обновляются независимо, что позволяет обучать сеть более эффективно, поэтому данная архитектура может выступать в качестве альтернативы архитектурам трансформера [2]. Также была показана возможность адаптации предложенной архитектуры для обнаружения аномалий в потоках текстового контента [4].

В данной работе предлагается новое решение поставленной задачи, основанное на рассмотренных архитектурах, и представляющее собой автокодировщик с использованием параллельной свёртки в качестве кодировщика и однослойного полносвязного декодировщика, который позволяет минимизировать потери при уменьшении размерности признакового пространства путем привлечения сравнительно небольшого объема вычислительных затрат. Также в работе для обнаружения аномалий предлагается использовать слой, основанный на нечеткой кластеризации, который позволяет автоматически определять уровень типичности каждого блока событий на основании информации о расстоянии от построенных сверточных признаков каждого блока до центра построенного кластера, позволяя при этом строить робастное решение, допускающее определенный процент выбросов в обучающей выборке. При этом слой нечёткой кластеризации позволяет ограничить минимальный размер кластера нормальных данных, что, в свою очередь, не приводит к построению кластера, состоящего всего из одной точки. Итоговая оценка нормальности блока событий оценивается как произведение степени типичности блока, полученной из слоя нечёткой кластеризации, на ошибку реконструкции, которая вычисляется как косинусная мера схожести результата декодирования и исходного векторного описания рассматриваемого блока событий. Иными словами, в процессе обучения сети на нормальных данных мы ищем центр распределения сверточных признаков, пытаясь при этом приблизить векторные представления рассматриваемых блоков к указанному центру и минимизируя ошибки кодирования сверточной частью сети. Предлагамая архитектура приведена на рис. 1 и 2.

3.1. Векторизация данных. В предлагаемом решении, как было сказано выше, рассматриваются непрерывные последовательности событий. При этом, к текстовому описания каждого события приминается единственная предобработка, связанная с удалением идентификаторов (последовательностей непробельных символов, содержащих цифры). После указанной предварительной обработки проводится токенизация по отдельным событиям и проводится их кодирование. Поскольку, как было сказано выше, объем уникальных событий невелик, сложные методы построения информативных векторных представлений текстовых данных (`Word2Vec`, `BERT` и т. д.) добавляют лишний шум к данным и ухудшают качество итоговых алгоритмов. Поэтому в данной работе для кодирования описаний событий предлагается использовать простой алгоритм `One-hot`.

В результате предварительной обработки и векторизации мы получаем представление описаний событий в виде потока `One-hot` векторов:

$$x = (x_1, x_2, \dots, x_i, \dots).$$

Здесь $x_i \in \mathbb{R}^m$ — вектор, описывающий событие с номером i, m — размер словаря. Далее события группируются в блоки по n событий, где размер блока n выбирается на основании особенностей рассматриваемых данных. Таким образом, на вход алгоритму построения модели данных

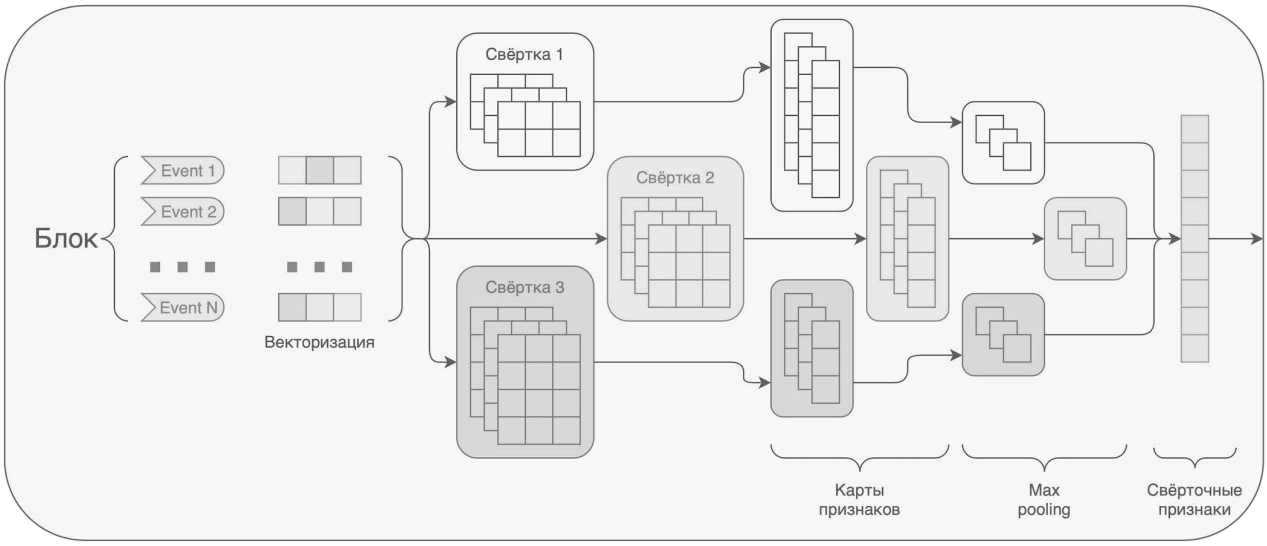

Рис. 1. Операция получения сверточных признаков

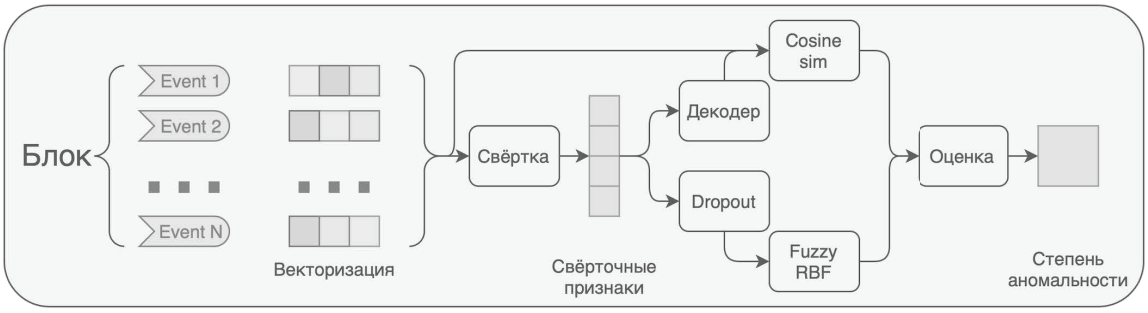

Рис. 2. Полная архитектура предлагаемого решения

подаются матрицы:

$$M = (M_1, M_2, \dots, M_i, \dots).$$

Здесь $M_i \in \mathbb{R}^{m*n}$ — матрица, описывающая блок с номером i.

3.2. Сверточные нейронные сети для построения признакового пространства. Для векторизации блоков событий и построения данных предлагается использовать операцию параллельной свертки, описанной в работе Ю. Ким [9].

Данная свертка представляет собой 3 независимых параллельных сверточных слоя, каждый из которых представляет собой отдельный сверточный слой, который выполняет преобразование последовательности группы признаков по определенному правилу. Выбирается некоторое значение $h \in \mathbb{N}$ — размер фильтра для свертки. Этот размер задается на этапе построения модели.

Далее рассмотрим отдельную матрицу M_i, которая будет передана одному сверточному слою. Каждый сверточный слой использует несколько независимых фильтров заданного размера, но с разными значениями весов. Тогда рассмотрим отдельный слой и обозначим через $\hat{c}_{i,j}$ карту признаков, которая получается после применения фильтра w_j указанного слоя к матрице M_i. Тогда выполняется следующее соотношение:

$$\hat{c}_{i,j} = M_i \circ w_i.$$

Здесь символ $\circ$ обозначает операцию свертки матрицы M_i фильтром w_j. Как и в работе Ю. Ким, в качестве фильтра используем матрицу размера $m * h$, т.е. $w \in \mathbb{R}^{m*h}$. Благодаря этому после

применения одного фильтра конкретного слоя получим карту признаков в виде вектора $\hat{c}_{i,j} \in \mathbb{R}^{n-h+1}$.

Далее к каждому значению в полученной карте признаков для указанного фильтра добавляется некоторое смещение и применяется нелинейная функция активации. В работе Ю. Ким [9] предлагалось использовать функцию `ReLU`. Однако она не позволяет учитывать отрицательные значения признаков, а также не осуществляет стандартизацию признаков, что ухудшает качество работы алгоритмов. В данной работе предлагается использовать модификацию указанной функции в виде `SELU` (см. [10]):

$$\texttt{SELU}(x) = \lambda * \begin{cases} x, x > 0 \\ \alpha * (e^x - 1),\ x \leqslant 0. \end{cases}$$

Здесь $\lambda, \alpha \in \mathbb{R}$ — константы нормализации (см. [10]).

После указанных преобразований для конкретного фильтра выполняется операция субдискретизации, в результате которой выбирается наиболее значимое (максимальное) значение в карте признаков. Следовательно, в результате комбинации свертки и субдискретизации для каждого фильтра получаем одно значение $\tilde{c}_{i,j} \in \mathbb{R}$ — сверточный признак для заданного фильтра.

После этого происходит конкатенация всех сверточных признаков для всех сверточных слоев и всех фильтров, в результате которой получается вектор признакового описания блока событий, который и можно рассматривать в качестве модели данных: $c_i \in \mathbb{R}^k$, где $k = \sum\limits_{p=1}^{n_c} s_p$, где $n_c = 3$ — количество сверточных слоев, s_p — количество фильтров в слое с номером p. Количество фильтров каждого слоя является гиперпараметром, который указывается на этапе построения модели.

3.3. Асимметричный автокодировщик для минимизации потерь информации при сокращении размерности признакового пространства. Анализ существующих работ показал, что для минимизации потерь информации при сокращении признакового пространства стоит использовать архитектуру декодировщика [2]. Таким образом, после сверточной части необходимо добавить слой декодировщика, который будет производить восстановление исходного векторного описания событий по построенным сверточным признакам.

При этом в нашей задаче не требуется точное восстановление первоначальных описаний событий. Ключевыми факторами при разработке декодировщика являются простота и скорость обучения. При этом важно, чтобы аномальные события имели большую по модулю ошибку восстановления, чем нормальные данные. Поэтому в данной работе предлагается использовать полносвязный однослойный декодировщик, а в качестве ошибки восстановления вычислять косинусную меру схожести, которая широко применяется в различных автокодировщиках для анализа текстовых данных:

$$\mathcal{L}_{\texttt{decomp}} = \cos(x_i, z_i).$$

Здесь z_i — восстановленный вектор исходного представления блока событий x_i, который получается в результате работы декодировщика.

В результате проведенных экспериментов было принято решение использовать в качестве декодировщика обычный полносвязный слой. При этом в процессе обучения сети происходит максимизация косинусной меры схожести. Иными словами, в процессе обучения модели данных мы строим информативное признаковое описание для каждого блока событий посредством применения сверточной архитектуры, при этом пытаясь восстановить первоначальные данные из построенных признаков с помощью полносвязного декодировщика.

3.4. Нечеткая кластеризация и выявление аномалий. Далее предлагается разработать алгоритм, позволяющий определять аномальное поведение на основании построенной модели данных. Как было сказано выше, для этих целей широко применяется подход сведения к задаче одноклассовой классификации. При этом наиболее эффективным является алгоритм, основанный на нечетком методе выявления аномалий в данных на основе эллиптической кластеризации, который был описан ранее [8].

В данной работе предлагается модифицировать описанный выше процесс для возможности применения с предлагаемым решением. Для этого было предложено встроить указанный метод в предлагаемое решение виде обучаемого слоя. Данный слой выполняет кластеризацию сверточных признаков, строя описание распределения нормальных данных. При этом нормальные данные должны соответствовать найденному распределению, поэтому необходимо минимизировать расстояние от каждого вектора обучающей выборки до найденного центра распределения. Для этих целей используются радиально-базисные функция активации. Обучающими параметрами данного слоя являются:

(i) координаты центра кластера a;
(ii) матрица ковариаций C (так как матрица является положительно определенной и диагональной, для эффективности построенного решения, можно представить эту матрицу в виде вектора неотрицательных параметров);
(iii) параметр μ.

В качестве функции активации используется функция, определяющая значение u_i для каждого объекта. В качестве ошибки будет использовано следующее значение, взятое из поставленной задачи оптимизации:

$$\mathcal{L}_{\texttt{unsupervised}} = \mu \sum_{i=1}^{N} u_i^m * ||x_i - a||_C^2.$$

Из этого следует, что можно доработать итоговый выход сети для заданного исходного векторного представления блока событий x_i следующим образом:

$$\texttt{Out}(x_i) = u_i^m * \mathcal{L}_{\texttt{decomp}}.$$

Таким образом, в качестве ошибки реконструкции вычисляется косинусная мера схожести первоначального и восстановленного представления блока событий. Чем больше это значение, тем меньше ошибка реконструкции. При этом мы обучаем предлагаемую модель на задаче одноклассовой классификации, в которой преобладающее большинство данных является нормальным. Поэтому в результате применения обученного автокодировщика к аномальным данным получаем в среднем меньшие значения косинусной меры схожести, чем в процессе применения модели к нормальным данным.

В результате работы сети мы получаем итоговую оценку нормальности анализированного блока событий на основании корректности построения признакового пространства и с учетом степени типичности объекта, полученную из RBF-слоя.

При этом в процессе обучения мы должны приблизить значение $\texttt{Out}(x_i)$ к единице. Следовательно, данную модель можно рассматривать как модель регрессии, которая должна получать в качестве отклика значение, близкие к 1. Тогда в качестве функции ошибки выходного слоя можно использовать следующую функцию:

$$\mathcal{L}_{\texttt{anom}} = \texttt{MSE}(\texttt{Out}(x_i), 1).$$

Значит, можно описать функцию ошибки, которая минимизируется в процессе обучения всей сети:

$$\mathcal{L} = \mathcal{L}_{\texttt{unsupervised}} + \mathcal{L}_{\texttt{anom}}.$$

3.5. Регуляризация. В ходе анализа существующих работ [4, 9], а также проведенных экспериментов с различными архитектурами, авторы пришли к выводу, что для избежания переобучения нужно использовать следующие два подхода:

(i) добавление $L2$-регуляризации для сверточного слоя, что позволяет ограничить значения весов сверточных слоев. Таким образом, итоговая функция ошибки будет выглядеть следующим образом:

$$\mathcal{L} = \mathcal{L}_{\texttt{unsupervised}} + \mathcal{L}_{\texttt{anom}} + \mathcal{L}_{l2};$$

(ii) использование слоя `Dropout` перед построением кластеризации. Это увеличивает качество обобщения сети для новых данных.

Анализ существующих работ показал, что стандартный алгоритм `Dropout` не позволяет добиться хороших результатов, так как он не сохраняет распределение исходных признаков после случайного удаления весов. Поэтому в данной работе предлагается использовать модификацию данного подхода, устраняющую указанный недостаток — `AlphaDropout` (см. [10]).

4. Экспериментальная оценка предлагаемого решения. Для экспериментальной оценки построенного решения предлагается провести подбор оптимальных гиперпараметров, а также сравнение данного подхода с рядом существующих классических подходов, таких как одноклассовый метод опорных векторов, методы нечеткой кластеризации. Также необходимо провести сравнение с более современными подходами, в частности, с архитектурой трансформера, использующего механизм самовнимания.

Как правило, для экспериментальной оценки эффективности работы того или иного алгоритма используются различные интегральные метрики качества. Одни из самых широко используемых метрик в данной задаче — это площадь под `ROC`-кривой (`AUC-ROC`, `PR-AUC`). Анализ существующих работ по тематикам, схожим с тематикой настоящей работы, показывает, что, как правило, для оценки качества алгоритма выявления аномалий при работе пользователя применяются именно эти метрики (см. [1, 4]). Поэтому они и будут использованы в настоящей работе для оценки эффективности предлагаемого алгоритма. При этом для проведения более достоверной оценки в данной работе предлагается провести бутстреппинг тестовой выборки и более точной оценки распределения предлагаемых метрик качества на основании вычисления медианного значения метрик, а также 1 и 3 квартиля по всем получаемым подвыборкам.

4.1. Используемые наборы данных. В целях проведения качественного экспериментального исследования было принято решения использовать следующие 2 набора данных, которые широко применяются для оценки решений схожих задач:

(a) `HDFS`: этот набор данных представляет собой поток текстовых описаний событий, происходящих на более чем 200 узлах `Amazon EC2`. Суммарно данный набор данных содержит порядка 11 млн записей об отдельных событиях. В описании каждого события содержится информация о привязке к некоторому блоку, который отражает привязку события к отдельной выполняемой программе в системе. В данном наборе данных присутствует разметка блоков событий на нормальные и аномальные (см. [20]).

(b) `BGL` (`Blue Gene/L`): этот набор данных содержит описания событий, происходящих на суперкомпьютерной системе `BlueGene/L`. Он содержит в себе порядка 5 млн записей об отдельных событиях. При этом для каждого события присутствует информация, является ли оно нормальным или аномальным. Всего в наборе присутствует порядка 350000 (7%) аномальных событий [16].

Далее для каждого набора данных будет приведено более подробное описание, а также будут сформулированы некоторые начальные установки для корректного проведения экспериментов.

4.2. Особенности применения предлагаемого алгоритма к рассмотренным наборам данных.

Векторизация данных и критерий аномальности. Как было сказано ранее, для предварительной обработки всех наборов данных используется удаление идентификаторов посредством регулярных выражений. Также на основании экспериментальных исследований было обнаружено, что в наборе данных `BGL` часто и то же событие дублируется несколько раз, поэтому было принято решение об очистке набора данных и сохранения только одного упоминания тех событий, информация о которых повторяется последовательно несколько раз. На этапе токенизации происходит разделение содержимого системных журналов по описаниям отдельных событий.

После выполнения предварительной обработки событий и удаления различных идентификаторов можно получить словарь уникальных событий. Ниже указана информация о получившихся словарях событий для каждого набора данных:

Таблица 1. Описание объемов выборок для наборов данных HDFS и BGL

Значение		HDFS	BGL
Тренировочная выборка		601576	541385
Валидационная выборка	Нормальные блоки	3367	4263
	Аномальные блоки	3367	4263
Тестовая выборка	Нормальные блоки	13471	17053
	Аномальные блоки	13471	17053
Число подвыборок в тестовой выборкеа		10	10

I. HDFS:
   (a) размер словаря: 75 событий;
   (b) примеры обработанных данных:
      (9) INFO dfsDataNodeBlockReceiver Exception writing block to mirror;
      (9) WARN dfsDataBlockScanner Adding an already existing block;

II. BGL:
   (a) размер словаря: 728 событий;
   (b) примеры обработанных данных:
      (i) RAS KERNEL INFO ciod Received signal;
      (ii) RAS KERNEL FATAL rts bad message header index greater than total.

Далее выполняется классическое One-hot кодирование всех событий на основании построенного словаря и порядкового номера каждого события в словаре.

В силу того, что размер входных данных должен быть фиксирован, необходимо группировать события в блоки фиксированного размера. Размер этих блоков и определение критерия аномальности зависит от особенностей конкретного набора данных:

1. В случае с данными HDFS события уже сгруппированы по номеру блока, в данных экспериментах было принято решение брать по 20 первых событий в каждом блоке (если в блоке было менее 20 событий, к нему добавлялось новое событие <EMPTY EVENT> столько раз, чтобы размер блока был равен 20). В качестве уровня аномальности блока использовалась соответствующая метка о наличии аномалии в блоке в исходном наборе данных.
2. В наборе данных BGL присутствует информация об аномальности отдельных событий. Поэтому предлагается решать задачу предсказания аномалий: все события объединяются в блоки по 5 событий с помощью скользящего окна. В качестве уровня аномальности каждого блока используется оценка аномальности того события, которое следует непосредственно после указанного блока.

Разделение данных на тренировочную, валидационную и тестовую выборки. Как было сказано выше, рассматривалось сведение поставленной задачи к задаче одноклассовой классификации. Поэтому в тренировочную выборку попадали только нормальные данные. Для определения оптимальных значений гиперпараметров было принято решение выделение сбалансированной валидационной выборки, включающей в себя как аномальные блоки событий, так и нормальные. На построенной выборке вычислялась метрика ROC-AUC на каждой эпохе обучения сети. Тестовая выборка также была составлена сбалансированно по классам, чтобы добиться более корректного вычисления метрик оценки качества работы алгоритмов. При этом, для более точной оценки тестовая выборка была случайным образом разделена на 10 сбалансированных подвыборок одинакового размера, для каждой из которых независимо вычислялись метрики качества. Размеры валидационной и тестовой выборок определялись исходя из количества аномалий в данных. Объемы тренировочной, валидационной и тестовой выборок указаны в таблице 1. Все значения указываются в количестве рассматриваемых блоков.

Построение модели данных. Для построения модели данных, как было сказано выше, используется сверточные слои (используется три слоя), в которых размер и количество фильтров зависит от размера блока данных. Кроме того, предлагается оценить решение на `GPU` с возможностью применения `mixed precision`, поэтому, по рекомендациям разработчиков, следует выбирать размер фильтров, кратный 8 (см. [15]). Поэтому в наборе данных `HDFS` предлагается использовать по 16 фильтров размеров 10, 11 и 12. В наборе данных `BGL` используется по 64 фильтра размеров 2, 3 и 4. Поскольку в качестве функции активации используется `SELU`, начальные значения для фильтров берутся из подходящего распределения `LecunNormal` (см. [10]), которое представляет собой усеченное нормальное распределение со средним значением 0 и стандартным отклонением, равным $\sqrt{1/N}$, где N — количество элементов в рассматриваемом тензоре.

Построение модели обнаружения аномального поведения. На этом этапе в обоих наборах данных предлагается выполнить объединение данных в один кластер, в качестве степени нечеткости используется значение $m = 2$. В качестве начального значения для параметра μ используется константа 0. В качестве начального значения для центра кластера используется непрерывное нормальное распределение со средним значением 0 и дисперсией 1. В качестве начального значения для ковариаций при вычислении расстояния Махалонобиса используется единичная матрица.

Регуляризация. В ходе экспериментальных исследований было выяснено, что значение $l2$-регуляризации сильно влияет на точность полученного решения и зависит от размера и количества используемых фильтров. Поэтому для набора `HDFS` в качестве оптимального значения константы регуляризации было выбрано 0,001, в наборе данных `BGL` — значение 0,9. Эти параметры также определялись на валидационном наборе данных.

4.3. Сравнение предлагаемого решения с классическими алгоритмами. Для оценки предлагаемого решения была проведена серия экспериментов по сравнению решения с существующими подходами, дающими лучший результат в похожих задачах: одноклассовый метод опорных векторов (`OCSVC`), нечеткая кластеризация (`Fuzzy`), трансформеры (`LogBERT`). При этом проведённые эксперименты показали, что в качестве алгоритма векторизации блоков событий для метода опорных векторов лучше всего подходит вектор, равный сумме `One-hot` векторов событий, входящих в блок («мешок событий»); для `Fuzzy` оптимальным стал подход, основанный на `TF-IDF`; подход `LogBERT` основан на архитектуре трансформера, который учитывает как индекс события в словаре, так и его позиция в блоке, поэтому для векторизации использовалась сумма векторов: случайно закодированный индекс события в словаре и векторное представления номера позиции, закодированное с помощью функции синусоиды. Также во всех алгоритмах осуществлялся подбор гиперпараметров на валидационной выборке. Рассматриваемые гиперпараметры и лучшие значения приведены ниже:

1. Метод опорных векторов:
   (a) функция активации ядра: радиально-базисная для обоих наборов;
   (b) параметр γ для ядровой функции: 0,1 для `HDFS` и 0,5 для `BGL`.
2. Метод нечеткой кластеризации:
   (a) степень нечеткости: 1,3 для обоих наборов;
   (b) процент исключений: 0,1 для `HDFS` и 0,15 для `BGL`;
   (c) параметр γ для ядровой функции: 0,3 для `HDFS` и 0,1 для `BGL`.
3. Метод `LogBERT`:
   (a) количество слоев типа «трансформер»: 2;
   (b) размерность представления токенов и внутреннего состояния: 50 и 256 соответственно;
   (c) коэффициент α в формуле вычисления итоговой ошибки;
   (d) процент замаскированных событий m в каждом блоке: 0,65 для `HDFS` и 0,5 для `BGL`;
   (e) число кандидатов g для определения степени аномальности предсказанного события: 6 в `HDFS` и 15 в `BGL`.

Результаты проведенных экспериментов представлены в таблице 2.

Таблица 2. Результаты проведенных экспериментов

Набор данных	Подход	ROC-AUC			PR-AUC		
		Медиана	Q1	Q3	Медиана	Q1	Q3
HDFS	OCSVC	0,833	0,826	0,837	0,828	0,821	0,832
	Fuzzy	0,858	0,851	0,862	0,853	0,847	0,856
	LogBERT	0,900	0,894	0,907	0,901	0,898	0,911
	FuzzyCNN	**0,973**	**0,971**	**0,974**	**0,970**	**0,969**	**0,972**
BGL	OCSVC	0,783	0,779	0,789	0,778	0,772	0,783
	Fuzzy	0,834	0,821	0,845	0,827	0,820	0,830
	LogBERT	0,908	0,901	0,912	0,898	0,893	0,901
	FuzzyCNN	**0,939**	**0,934**	**0,940**	**0,921**	**0,916**	**0,926**

Таким образом, во всех проведенных экспериментах на рассмотренных наборах данных предлагаемое решение позволяет добиться лучшего результата.

5. Заключение. В данной работе предлагается новый подход к задаче обнаружения аномальных событий в данных системных журналов на основании архитектуры автокодировщика с использованием параллельного свертки для кодирования поступающих блоков событий, полносвязного декодировщика для минимизации потерь при кодировании, а также нового слоя, основанного на нечеткой кластеризации сверточных признаков, выход которого перемножается на ошибку реконструкции декодировщика, что позволяет определить совокупную степень аномальности каждого блока событий. Предлагаемое решение позволяет эффективно строить векторные представления потока сложно структурированных данных системных журналов с учетом внутренних взаимосвязей между событиями.

Предложенное решение позволяет значительно уменьшить временные затраты существующих решений поставленной задачи путем применения параллельной свертки. Кроме того, для восстановления исходного векторного представления последовательности событий достаточно применить обычный полносвязный слой, который также не требует значительных вычислительных затрат. Следовательно, предложенное решение может выступать эффективной альтернативой существующим архитектурам Transformer, позволяющим добиться лучших результатов в поставленной задаче.

СПИСОК ЛИТЕРАТУРЫ

1. *Chandola V., Banerjee A., Kumar V.* Anomaly detection: A survey// ACM Comp. Surv. (CSUR). — 2009. — *41*, № 3. — P. 1–58.
2. *Chollet F.* Deep Learning with Python. — Simon and Schuster, 2021.
3. *Girolami M.* Mercer kernel-based clustering in feature space// IEEE Trans. Neural Networks. — 2002. — *13*, № 3. — P. 780–784.
4. *Gorokhov O., Petrovskiy M., Mashechkin I.* Convolutional neural networks for unsupervised anomaly detection in text data// Int. Conf. on Intelligent Data Engineering and Automated Learning. — Cham: Springer, 2017. — P. 500–507.
5. *Guo H., Yuan S., Wu X.* Logbert: Log anomaly detection via bert// Proc. 2021 International Joint Conf. on Neural Networks. — IEEE, 2021. — P. 1–8.
6. *He S. et al.* Experience report: System log analysis for anomaly detection// Proc. 2016 IEEE 27th Int. Symp. on Software Reliability Engineering. — IEEE, 2016. — P. 207–218.
7. *Hotho A., Nurnberger A., Paas G.* A brief survey of text mining// J. Language Techn. Comput. Linguistics. — 2005. — *20*, № 1. — P. 19–62.

8. *Kazachuk M. et al.* Novelty detection using elliptical fuzzy clustering in a reproducing kernel Hilbert space// Proc. 19th Int. Conf. "Intelligent Data Engineering and Automated Learning (IDEAL-2018). Part 2 (Madrid, Spain, November 21–23, 2018). — Springer, 2018. — P. 221–232.
9. *Kim Y.* Convolutional neural networks for sentence classification/ arXiv: 1408.5882 [cs.CL].
10. *Klambauer G. et al.* Self-normalizing neural networks// Adv. Neural Inform. Process. Syst. — 2017. — *30*. — P. 971–980.
11. *Le V. H., Zhang H.* Log-based anomaly detection with deep learning: How far are we?// Proc. 44th Int. Conf. on Software Engineering, 2022. — P. 1356–1367.
12. *Ma J. et al.* Automatic parsing and utilization of system log features in log analysis: A Survey// Appl. Sci. — 2023. — *13*, № 8. — P. 4930.
13. *Manevitz L. M., Yousef M.* One-class SVMs for document classification// J. Machine Learn. Res. — 2001. — *2*. — P. 139–154.
14. *Mi H. et al.* Toward fine-grained, unsupervised, scalable performance diagnosis for production cloud computing systems// IEEE Trans. Paral. Distr. Syst. — 2013. — *24*, № 6. — P. 1245–1255.
15. *Micikevicius P. et al.* Mixed precision training/ arXiv: 1710.03740 [cs.AI].
16. *Oliner A., Stearley J.* What supercomputers say: A study of five system logs// Proc. 37th Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks. — IEEE, 2007. — P. 575–584.
17. *Petrovskiy M. I.* Outlier detection algorithms in data mining systems// Program. Comput. Software. — 2003. — *29*. — P. 228–237.
18. *Ryciak P., Wasielewska K., Janicki A.* Anomaly detection in log files using selected natural language processing methods// Appl. Sci. — 2022. — *12*, № 10. — P. 5089.
19. *Vaswani A. et al.* Attention is all you need// Adv. Neural Inform. Process. Syst. — 2017. — *30*. — P. 5998–6008.
20. *Xu W. et al.* Detecting large-scale system problems by mining console logs// Proc. ACM SIGOPS 22nd Symp. on Operating Systems Principles, 2009. — P. 117–132.

ДЕКЛАРАЦИЯ АВТОРОВ

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Финансирование. Авторы заявляют об отсутствии финансовой поддержки от каких-либо организаций или частных лиц.

Финансовые интересы. Авторы заявляют об отсутствии подлежащих раскрытию финансовых или нефинансовых интересов, связанных с публикуемым материалом.

Горохов Олег Евгеньевич (Gorokhov Oleg Evgenievich)
Московский государственный университет имени М. В. Ломоносова
(M. V. Lomonosov Moscow State University, Moscow, Russia)
E-mail: owlman995@gmail.com

Петровский Михаил Игоревич (Petrovskii Mikhail Igorevich)
Московский государственный университет имени М. В. Ломоносова
(M. V. Lomonosov Moscow State University, Moscow, Russia)
E-mail: michael@cs.msu.su

Машечкин Игорь Валерьевич (Mashechkin Igor Valerievich)
Московский государственный университет имени М. В. Ломоносова
(M. V. Lomonosov Moscow State University, Moscow, Russia)
E-mail: mash@cs.msu.su