DOI: 10.15514/ISPRAS-2026-38(3)-25

Anomaly Detection in Computer System Logs Using semi-supervised Learning and Natural Language Processing

V.A. Kiriachek, ORCID: 0009-0002-9692-0225 <w.a.kiryachok@mail.ru>
S.I. Salpagarov, ORCID: 0000-0002-5321-9650 <salpagarov-si@rudn.ru>
RUDN University,
6, Miklukho-Maklaya st., Moscow, 117198, Russia.

Abstract. The detection of anomalies in computer system logs is crucial for maintaining reliable technological infrastructures. This study introduces a novel approach combining semi-supervised learning with Natural Language Processing to analyze log files for early identification of potential system failures. The methodology employs a specialized log parser based on semantic graphs alongside context-independent embedding models for text vectorization, focusing on collective rather than point anomalies. Experiments were conducted on both the public HDFS dataset and a proprietary Vertica database dataset containing over 1 113 million logs. Results demonstrate that the obtained solution based on autoencoders with convolutional layers can effectively detect system anomalies when paired with appropriate preprocessing techniques. The approach achieved impressive performance metrics on the HDFS dataset, particularly when using TF-IDF token weighting, with a Fault Detection Rate of 0.982 and ROC AUC of 0.811. Additionally, testing on the Vertica dataset successfully identified anomalous periods preceding system failures. The findings indicate that predictive maintenance approaches traditionally applied to technical equipment can be successfully adapted for computer systems, enabling proactive intervention before critical failures occur and potentially reducing the significant costs associated with system downtime.

Keywords: anomaly detection; log analysis; semi-supervised learning; natural language processing; predictive maintenance; TF-IDF vectorization.

For citation: Kiriachek V.A., Salpagarov S.I. Anomaly detection in computer system logs using semi-supervised learning and natural language processing. Trudy ISP RAN/Proc. ISP RAS, vol. 38, issue 3, part 2, 2026, pp. 133-148. DOI: 10.15514/ISPRAS-2026-38(3)-25.

Обнаружение аномалий в журналах компьютерных систем на основе полуобучения и обработки естественного языка

В.А. Кирячёк, ORCID: 0009-0002-9692-0225 <w.a.kiryachok@mail.ru>
С.И. Салпагаров, ORCID: 0000-0002-5321-9650 <salpagarov-si@rudn.ru>
Российский университет дружбы народов имени Патриса Лумумбы,
Россия, 117198, г. Москва, ул. Миклухо-Маклая, д. 6.

Аннотация. Обнаружение аномалий по журналам событий компьютерных систем имеет определяющее значение для поддержания надёжности технологических инфраструктур. В этом исследовании представлен новый подход, сочетающий машинное обучение с частичным привлечением учителя вместе с обработкой естественного языка для анализа журналов событий, направленный на раннее выявление потенциальных сбоев в компьютерных системах. В исследовании используется специализированный парсер журналов событий, основанный на семантических графах, наряду с контекстно-независимыми моделями векторного представления текста, с фокусом на коллективных, а не точечных аномалиях. Эксперименты проводились как на общедоступном наборе данных HDFS, так и на собственном из базы данных Vertica, содержащем более 1 113 миллионов журналов событий. Результаты показывают, что полученное решение, основанное на автокодировщиках со свёрточными слоями, может эффективно обнаруживать системные аномалии в сочетании с соответствующими методами предварительной обработки. Подход достиг впечатляющих результатов на наборе HDFS, особенно при использовании взвешивания токенов с помощью TF-IDF, с метриками Fault Detection Rate равной 0,982 и ROC AUC равной 0,811. Кроме того, тестирование на базе данных Vertica успешно выявило аномальные периоды, предшествующие системным сбоям. Результаты показывают, что подходы предиктивной диагностики, традиционно применяемые к техническому оборудованию, могут быть успешно адаптированы для компьютерных систем, позволяя проводить профилактическое вмешательство до возникновения критических сбоев и потенциально снижая значительные затраты, связанные с простоем системы.

Ключевые слова: обнаружение аномалий; анализ журналов событий; машинное обучение с частичным привлечением учителя; обработка естественного языка; предиктивная диагностика; TF-IDF векторизация.

Для цитирования: Кирячёк В.А., Салпагаров С.И. Обнаружение аномалий в журналах компьютерных систем на основе полуобучения и обработки естественного языка. Труды ИСП РАН, том 38, вып. 3, часть 2, 2026 г., стр. 133–148 (на английском языке). DOI: 10.15514/ISPRAS–2026–38(3)–25.

1. Introduction

Monitoring and complex analysis of events in computer systems is a critical component of ensuring the reliability, security, and continuity of modern technological infrastructures. The main source of information about system processes is log files – complex text documents containing a detailed record of events from normal operation to critical failures and abnormal situations. In the era of digital transformation, computer systems generate unprecedented amounts of data. Millions and billions of log lines are created every day, with an enormous variety of sources, formats, and contexts. Traditional methods of manually analyzing such huge amounts of information are becoming not just time-consuming, but practically impossible even for highly skilled teams of system administration specialists.

The central challenge of modern monitoring systems is the timely detection of anomalies and potential risks of equipment failure. Predictive maintenance (PdM) is changing the classic reactive approach to infrastructure maintenance, enabling a shift from remediation to early problem prevention.

The importance of systems for business processes cannot be overestimated. For example, unplanned database downtime can instantly paralyze the entire corporate monitoring system, management analytics, and operational processes. In manufacturing ecosystems, such failures can lead to

multimillion-dollar losses and disrupt entire process chains. According to recent studies, the average cost of downtime in large organizations has reached \$9000 per minute [1]. In companies with a higher level of risk, such as finance and healthcare, downtime in certain situations can exceed \$5 million per hour - and this is without considering potential fines or penalties [1]. The main causes of computer system downtime include equipment failures, cyberattacks, and network outages. Outdated infrastructure and lack of regular maintenance also contribute to unplanned system shutdowns. These downtimes can be prevented by investing in reliable infrastructure and using IT system monitoring tools that help identify problems before they occur.

The purpose of this work is to explore the possibility of applying predictive maintenance methods (typically used for technical equipment) to computer systems. This will be accomplished by developing specialized software for log analysis, which will allow for early detection of anomalies and prevention of system failures.

There are several approaches to anomaly detection in computer systems. According to the recognition modes, the approaches are divided into Supervised Anomaly Detection, Unsupervised Anomaly Detection, and semi-supervised Anomaly Detection [2]. The following approaches are distinguished by anomaly detection methods: classification, clustering, statistical analysis, nearest neighbor algorithm, spectral, and hybrid methods [2].

Intelligent log analysis systems based on advanced machine learning and Natural Language Processing (NLP) technologies are becoming the leading solution . These systems are qualitatively transforming the approach to system monitoring:

- automatic transformation of unstructured text data into ordered information arrays;
- intelligent search for latent patterns and potential anomalies;
- generation of proactive analytical reports with a high degree of accuracy;
- creation of intuitive early warning systems for critical failures;
- continuous machine learning based on accumulated historical data.

Such solutions not only optimize the work of technical services, but also become a strategic tool for digital transformation, allowing companies to significantly increase the resilience and efficiency of their technology ecosystems.

Having a certain structure, log files contain information about various system events such as errors, warnings, and other incidents. They record the time and date of the event, as well as its type or level of importance indicated by special tags (e.g., INFO, ERROR, FATAL). In addition, logs contain a significant amount of volatile data, including hash values, process IDs, network addresses, etc. This data can be generated dynamically and, as a rule, is not repeated in future records, which requires special preprocessing techniques to normalize and depersonalize it before further analysis.

2. Related Works

Many parsers built on different architectures such as Drain [3], Spell [4] and others have been discussed in the literature. Nevertheless, in order to meet the requirements of production tasks imposing additional constraints in the form of the need for integration into existing software and the possibility of flexible customization of individual system components, it was decided to develop our own specialized log parser.

Anomalies in computer systems are understood as events characterized by outlier values of their features and sharply contrasting with the typical operating modes of such systems during their normal operation. Anomalous behavior of systems is often rare and unpredictable, deviating from established patterns based on previous observations. Therefore, the developed anomaly detection approach is based on semi-supervised learning techniques together with context-independent embedding models such as fastText [5] with TF-IDF weighting [6], which do not require labeling

the training data. It is only necessary to know the periods of normal uninterrupted operation of the system.

The analysis of existing neural network-based solutions allows us to identify many different approaches to solving anomaly detection problems in computer systems, in which a significant proportion consists of supervised machine learning methods, such as LogRobust [7], CNN [8], as well as semi-supervised methods, such as DeepLog [9], LogAnomaly [10], LogBERT [11], PLELog [12], LogGPT [13], LogLLM [14], FastLogAD [15] or Unsupervised methods, such as Logsy [16]. While this paper focuses on semi-supervised learning for computer system logs, it's worth noting that Unsupervised deep learning methods have shown promising results in other anomaly detection domains, such as in the detection of abnormal electrocardiograms [17].

In the above articles, the main datasets used are HDFS [18], BGL [19] and Thunderbird [19], which range in size from several million to several tens of millions of logs. The logs in them are labeled into abnormal and normal, hence the choice of quality metrics to evaluate the algorithms - binary classification metrics such as Precision, Recall and F1-score. Timestamps do not participate in the calculation of metrics in any way, but it is important to identify abnormal periods of operation, thereby preventing the development of major anomalies leading to failures. Thus, this paper proposes grouping logs by timestamps to analyze the whole system rather than individual logs [20]. For factory equipment, there is software that solves the above-described pool of issues, for example, GE SmartSignal [21], but creating such software for logs is a promising task and the purpose of this work.

Recurrent Neural Networks [7, 9-10], Convolutional Neural Networks [8], Transformers [11-12], and Large Language Models [14-15] are among the most widely used machine learning architectures for anomaly detection. Unsupervised approaches that do not require labeling, such as Autoencoders [22-23], and classical methods like One Class SVM [24], Isolation Forest [25], and Local Outlier Factor [26], are also effective. Metric-based methods, including k-Nearest Neighbors with Euclidean and Manhattan metrics [20], and the Mahalanobis metric in the Elliptic Envelope algorithm [27], are commonly applied for anomaly detection [7, 9-10]. Similarly, generalized metrics, such as Euclidean-Mahalanobis, excel in pattern recognition tasks [28], demonstrating their versatility in data analysis. In this paper, we implemented classical machine learning methods, Autoencoders, and metric-based approaches.

3. Methods and Implementation for Anomaly Detection

3.1 Anomaly detection problem formulation

There are two approaches to solving the anomaly detection problem shown in Fig. 1 [2]. In the first, point-based approach, each log is considered as a separate vector, for which it is necessary to determine the class: abnormal or not. This approach is common in the literature, it is easy to calculate binary classification metrics such as F1-score for it, and the quality of existing solutions for open datasets tends to 1 [14]. The second approach considers not single anomalies, but group or collective ones that occur as a sequence of time points when there are no normal states between the beginning and the end of the anomaly.

Although log anomaly detection is actively researched, the vast majority of studies focus on point or contextual anomalies – detecting individual anomalous records or fixed-length sequences. Specialized methods aimed at identifying collective anomalies in aggregated log time blocks are extremely rare in the literature. Standard comparative studies and benchmarks do not evaluate methods from this perspective. Consequently, there is no established set of SOTA methods or generally accepted metrics for direct comparison in our specific problem setting.

Thus, the paper proposes to solve the problem for collective anomalies because in reality, single abnormal logs do not pose a serious threat and can rarely lead to serious consequences, unlike group

anomalies, which are an indication of serious problems in computer systems or factory equipment that led to breakdowns.

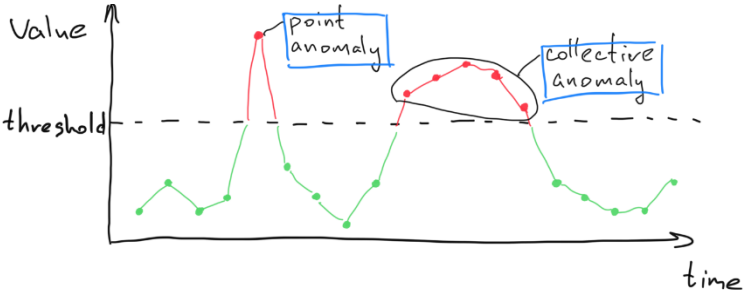

Fig. 1. Point and collective anomalies.

3.2 Description of the raw data format

The study attempted to use popular labeled datasets HDFS, BGL and Thunderbird, as well as a proprietary dataset from a large IT company's Vertica database [29].

First, open datasets were analyzed for the possibility of using them to detect group anomalies. The HDFS dataset contains 11,175,629 labelled logs. The logs were grouped into minute intervals to identify group anomalies. As can be seen in Fig. 2, the anomalies form clear groups and, if the threshold is set correctly, the HDFS dataset can be used for our task.

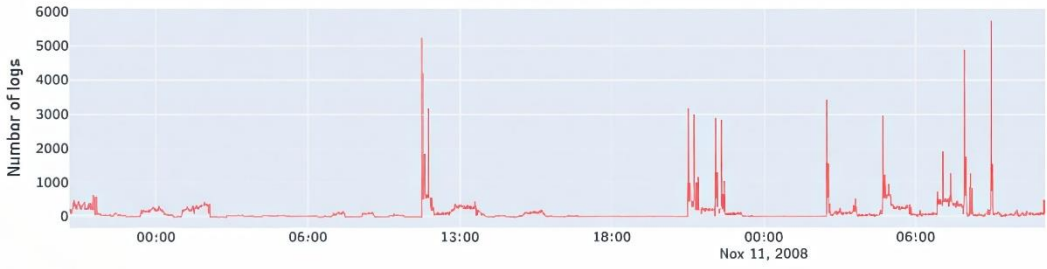

Fig. 2. Number of anomalies in minute intervals for the HDFS dataset.

The BGL dataset contains 4,747,963 labelled logs, which, after grouping, have the form shown in Fig. 3. As can be seen from the figure, the anomalies are point-like, so the dataset is not suitable for our task.

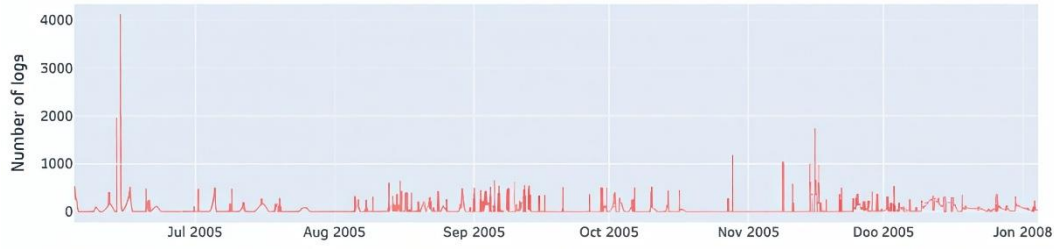

Fig. 3. Number of anomalies in minute intervals for the BGL dataset.

The Thunderbird dataset contains 211,212,192 labelled logs, which after grouping have the form shown in Fig. 4. Several collective anomalies are visible in the figure, the situation is complicated by the large number of single anomalies, so for now the use of Thunderbird is a focus for further research.

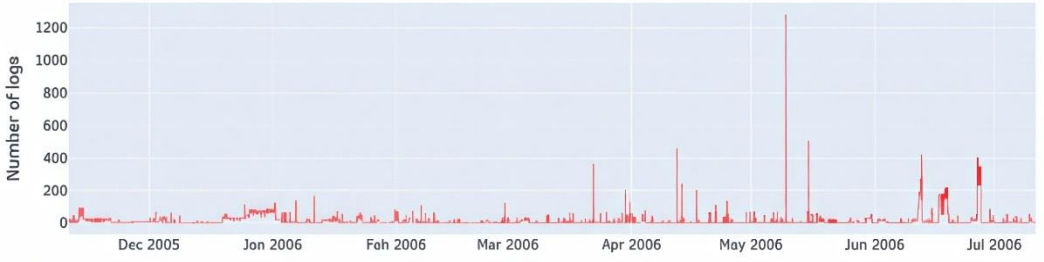

Fig. 4. Number of anomalies in minute intervals for the Thunderbird dataset.

Proprietary dataset, compiled based on the Vertica database, does not have explicit labeling, but there are two known periods when the database failed. It is not possible to calculate metrics for such a case, but it is possible to observe how the database behaved before the failure [20].

The 2-month training period, in which there were no failures, contains about 830 million logs. The first 7-day test period with a failure consists of about 135 million logs, and the second 10-day test period with a failure consists of about 148 million logs. Thus, this dataset is the largest of the analyzed datasets.

In all datasets, logs have a characteristic structure. Examples of several events from the HDFS dataset are shown in Fig. 5. First, there is information about the date and time of the event, the event tag, followed by the text of the event itself, containing variable information, i.e., IDs, hosts, ports, etc.

To transform the log stream into structured data suitable for model training, the source records were aggregated into sequential fixed-length time blocks. The key parameter of this procedure is the aggregation window length.

```
081110 022804 6189 INFO dfs.DataNode$DataXceiver: 10.250.9.207:50010 Served block blk_-3816021116470515763 to /10.250.9.207
081110 022804 6201 INFO dfs.DataNode$DataXceiver: 10.251.71.68:50010 Served block blk_-1320416988172655646 to /10.251.65.203
081110 022804 6245 INFO dfs.DataNode$DataXceiver: 10.251.30.85:50010 Served block blk_3726191010151813407 to /10.250.11.85
081110 022804 6292 INFO dfs.DataNode$DataXceiver: 10.251.70.37:50010 Served block blk_-4480983579096894108 to /10.251.199.86
081110 022804 6356 INFO dfs.DataNode$DataXceiver: 10.251.67.4:50010 Served block blk_-6530421042024996384 to /10.251.106.214
081110 022804 6373 INFO dfs.DataNode$DataXceiver: 10.251.127.191:50010 Served block blk_-860353257544297806 to /10.251.38.214
```

Fig. 5. Example of HDFS dataset raw log data.

As previously demonstrated, a threshold of one minute was set in this study. This choice was motivated by the following considerations:

- Preliminary analysis of anomaly patterns in Vertica logs revealed that many query-related incidents (e.g., cascading timeouts or syntax errors) develop and resolve within intervals close to a minute;
- To ensure comparability with earlier work [20], which substantiated the effectiveness of this window for Vertica, we retained this parameter;
- A 60-second window provides a balance between granularity (the ability to capture a short but significant event) and feature stability (each block contains enough records to compute informative statistics such as error rate or SQL pattern diversity).

As demonstrated in earlier work [20], reducing the window (to 0.2 minutes) can increase the system's sensitivity to short-lived anomalies and improve the accuracy of incident localization. However, this will lead to an increase in the number of blocks, increased computational costs for processing them, and a potential increase in data noise due to the low statistical content of each block. Increasing the window (to 5 minutes) can facilitate the detection of collective anomalies, which manifest themselves not as a sharp spike, but as a slow drift in system characteristics. This approach could potentially allow the BGL dataset, where some anomalies are distributed, to be used for testing. However, critical events may be distributed within intervals and be lost amid normal activity.

3.3 Description of the proposed approach

Among the features typical for logs of any database, not just the one used by Vertica, it is worth noting the presence of SQL queries in the logs themselves. This data can be useful, because a sub optimally written query can lead to problems in the database and even its failure. However, using existing popular log parsers (such as Drain, Spell) in their default configurations proved unsuitable for our task for two main reasons. First, these algorithms often treat quoted text or long string sequences as noise, replacing them with a single catch-all token (e.g., <*>), which leads to a complete loss of the semantics of SQL queries. Second, even if a query is saved as a single token, the internal structure (e.g., key WHERE and JOIN clauses, table and column names) remains inaccessible for analysis, which is critical for identifying performance issues. Thus, existing solutions, while effective for clustering similar system events, are unable to handle highly variable logic containing structured data. This led to the development of our own log parser, the architecture of which is based on a semantic graph.

The fundamental difference between our approach and hard clustering methods (like Drain) is the construction and subsequent context-sensitive collapse of a semantic graph. Instead of fixing the positions of static words and replacing everything else with wildcard tokens, our algorithm constructs a directed graph during the training phase, where the nodes are lexemes (words, numbers, special characters), and the edges reflect the probabilistic relationships between them in the sequence. A key step is the graph "collapse", in which high-frequency hub nodes (e.g., error codes, table names, etc.) are replaced with special tokens. Critically, the choice of a specific token is determined not only by the lexeme itself but also by its position in the graph (context). This allows us to preserve the semantic structure of the query, not just its template. The general scheme of the proposed approach is illustrated in Fig. 6 [20].

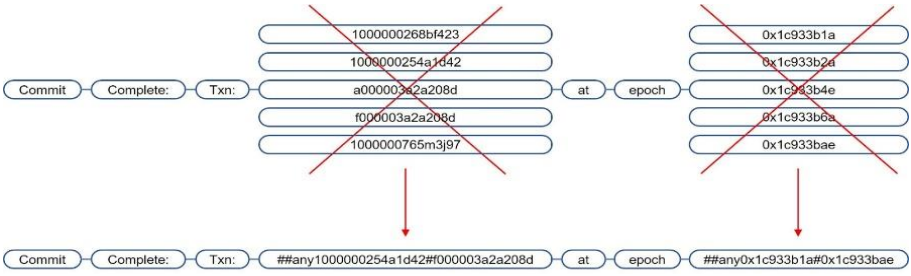

Fig. 6. Schematic overview of the proposed log parser.

In this study, a direct empirical comparison of parsing accuracy with alternatives proved methodologically flawed for a fundamental reason: the target outputs of the parsers are fundamentally different. Standard parsers are designed to aggressively cluster logs into abstract patterns, removing specific data (including SQL). Their benchmark accuracy is measured on datasets consisting of system messages (e.g., HDFS), where such aggregation is the goal. In our case, the goal is not removal, but rather the preservation and structuring of SQL queries for subsequent semantic analysis. Thus, if we fed our logs containing SQL as input to a standard parser, its "correct" output would be complete removal of queries, which is unacceptable for our task. Comparing the accuracy of a method that preserves information with a method that discards this information by design is meaningless. Therefore, the validation of our approach is qualitative and practical: we demonstrate that the proposed method solves a problem that cannot be solved by existing out-of-the-box tools.

After preprocessing the data using the parser, textual information is converted to numeric using the context-independent embedding models, such as following the procedure described in [20]. To ensure compatibility with distance-based algorithms like k-NN, the resulting vectors were normalized using L2-normalization, preserving the relative distances between data points while standardizing their magnitudes.

Since logs typically contain many repeated structural elements (e.g., function names or error codes) where the semantics of individual words rather than the context is important, context-independent embedding models like fastText [5], Word2Vec [30] or Glove [31] are ideal for log vectorization because they produce single, stable vector representations for each word, which is critical for anomaly detection. In contrast, BERT [32] produces different vectors for the same word in different contexts, which is redundant for structured logs and makes embeddings very difficult to compute and compare.

Since the logs can receive tens of thousands of individual events per minute, they must first be grouped. In the work, this was done using a sliding window with a step equal to the window size. Thus, logs are divided into equal time intervals with different number of events falling into them. The procedure for averaging vectors within a single time interval is as follows: first, the vector of one event is calculated as the arithmetic mean of all tokens included in it, and then the arithmetic mean of all events in one interval is found. As a result, one time interval corresponds to one vector of the dimension that is set when training the fastText model. The tokens found in the logs are not equal, so the ability to weigh them using TF-IDF has also been added. To do this, for each token within a time interval, its TF-IDF value is calculated and then this value is multiplied by its vector. The same scheme was applied to whole log patterns, which were combined into patterns after the parser. This approach allows one to reduce the importance of common tokens (log patterns) and increase it for rare ones. The general scheme of vectorization and averaging is shown in Fig. 7.

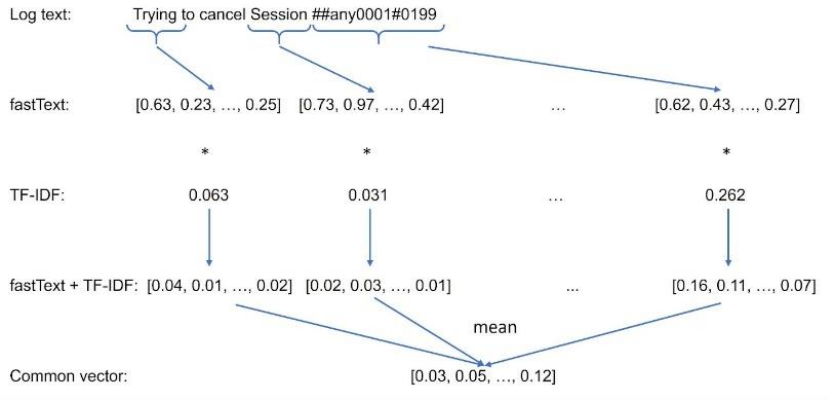

Fig. 7. Scheme of log vectorization.

This approach is a simplification, as it doesn't take into account positional information or the order of messages within a time interval. However, this simplification is intentional and justified by the specifics of our task and data. First, after processing by the parser, heterogeneous logs are reduced to a limited set of semantic patterns. The order of occurrence of such similar events within a minute window is often less meaningful for anomaly detection than their aggregate statistics (frequency, composition). Second, our goal is to identify anomalous system states over an interval, characterized, for example, by a spike in errors of a certain type or the appearance of rare SQL patterns. Averaged embedding effectively accumulates these statistical shifts, serving as a kind of "fingerprint" of the system state over the time window. Finally, this approach provides computational efficiency and robustness when dealing with long, noisy sequences, allowing the model to focus on meaningful changes in the distribution of events rather than their precise timing within a window.

After the vectors for each time interval have been obtained for both the training and test periods, you can start the anomaly detection task. In this formulation, there are known time periods when there is confidence that the computer system is operating without anomalies and there is no other data labeling. Usually, as mentioned earlier, semi-supervised learning approaches are used in such cases. Among the methods tested in the work the following algorithms should be noted: One Class

SVM, Isolation Forest, Local Outlier Factor, Elliptic Envelope, k-Nearest Neighbours, and Autoencoders.

3.4 Quality Metrics

To calculate metrics, it is necessary to switch from the number of abnormal logs in minute intervals to binary labeling using a set threshold. Fig. 8 shows such a labeling for the HDFS dataset, based on the graph in Fig. 2 with the selected threshold of 40. Thus, all minute intervals containing fewer than 40 anomalies are considered normal, and those containing more than 40 are considered abnormal. The threshold value of 40 was chosen based on the overall distribution of anomalies in the intervals but is not rigidly fixed and can be changed later if necessary.

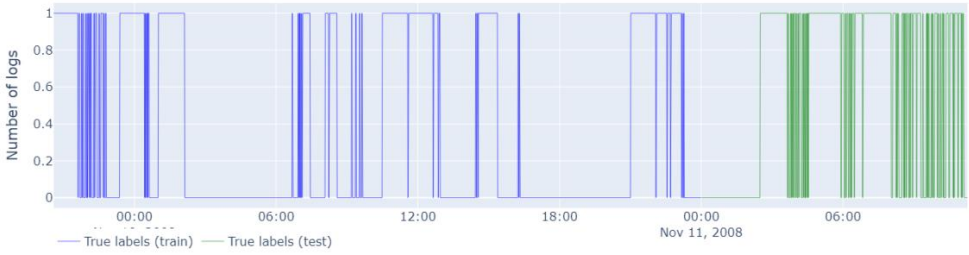

Fig. 8. Binary labeling for the HDFS dataset (blue is the training period, green is the test period).

A good anomaly detector should identify the anomaly zones most fully, and at the same time should not give many false positives. The following metrics were chosen as binary classification metrics:

- FDR (Fault Detection Rate) or TPR (True Positive Rate) or Recall (Completeness);
- FAR (False Alarm Rate) or FPR (False Positive Rate).

$$FDR = TPR = Recall = \frac{TP}{TP+FN}, \quad (1)$$

where TP are true positives, FN are false negatives.

$$FAR = FPR = \frac{FP}{FP+TN}, \quad (2)$$

where FP are false positives, TN are true negatives.

This choice of metrics is motivated by the fact that FDR shows the proportion of coverage of abnormal zones and FAR shows the proportion of false positives.

In addition, the ROC AUC (the area under the ROC curve), which allows one to evaluate quality without setting a threshold value, as well as the Precision and F1-score, were calculated.

4. Results

4.1 Results for the open HDFS dataset

Autoencoders with convolutional layers were chosen as the basic anomaly detection algorithm due to their relative ease of implementation, their ability to efficiently detect complex hierarchical patterns and local anomalies in sequential data (e.g. logs) by compressing them into a latent representation and accurately identifying outliers at the expense of high reconstruction error, and their robustness to small shifts, since an anomaly can occur anywhere in the sequence and convolutional layers are invariant to the position of the pattern, making the model more robust. Fig. 9 shows anomaly graphs for the test period of the HDFS dataset, calculated after vectorization of logs by the fastText algorithm together with TF-IDF weighting only logs (Fig. 9a), only tokens (Fig. 9b), logs and tokens simultaneously (Fig. 9c).

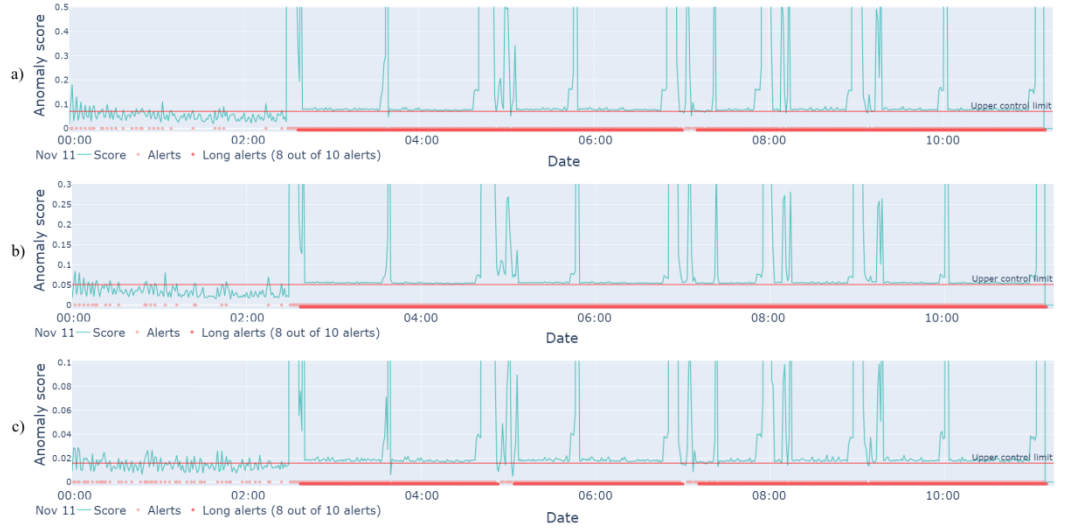

Fig. 9. Test period anomaly graphs of HDFS dataset with TF-IDF weighting: a) logs only; b) tokens only; c) logs and tokens simultaneously.

The pink color at the bottom of the graphs indicates exceeding the Upper Control Limit (UCL) set based on the training data. The red color at the bottom of the graphs indicates long-term exceeding of the UCL for filtering single triggers. They are calculated using a "sliding" window, which considers an anomaly to be triggered only when 8 out of 10 threshold exceedances occur.

Fig. 10 shows the binary labels for all three approaches (blue indicates the actual labeling and green the calculated one) corresponding to the red dots in Fig. 9.

Fig. 10. Actual (blue) and calculated (green) binary labels (0 for normal, 1 for anomalous) for the test period of the HDFS dataset with TF-IDF weighting across three scenarios: a) analysis using logs only; b) analysis using tokens only; c) analysis using both logs and tokens simultaneously.

Table 1 shows all the calculated metrics for the test period. It can be seen that TF-IDF weighting of tokens only turned out to be the best of the three approaches.

As Figure 10 and Table 1 show, the False Alarm Rate (FAR) metric reaches 44% on on the public HDFS dataset. This high value is due to the HDFS dataset's inherent design for detecting individual anomalous records (point anomalies), not collective patterns. Using it for our task required artificial

relabeling, which inevitably introduced noise and increased the false positive rate during time block analysis. Importantly, this drawback is not observed on real production logs from the Vertica database, for which the method was developed. This confirms that the identified issue is specific and related to the imperfect fit of the HDFS dataset to our task, rather than a fundamental flaw in the approach.

Table 1. Calculated metrics for the test period of the HDFS dataset.

Approach	FDR (Recall)	FAR	Precision	F1-score	ROC AUC
a) TF-IDF weighting of logs only	0.960	0.444	0.754	0.845	0.761
b) TF-IDF weighting of tokens only	0.982	0.444	0.759	0.856	0.811
c) TF-IDF weighting of logs and tokens simultaneously	0.932	0.444	0.749	0.831	0.718

Fig. 11 shows a comparison of the ROC curves, the areas under which are shown in Table 1.

Fig. 12 shows graphs comparing anomaly detection methods for the model with TF-IDF weighting of tokens only (Fig. 9b) as the best-performing approach. All the graphs clearly show the occurrence of a series of anomalies, only the shapes of the peaks and the scale differ.

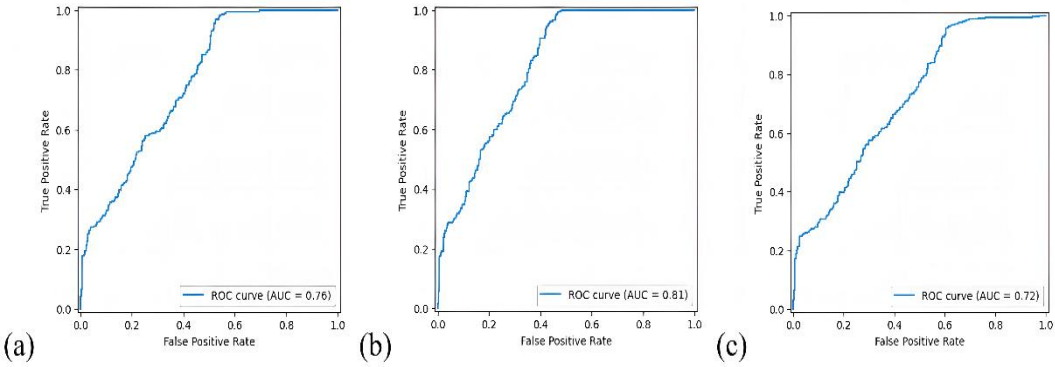

Fig. 11. ROC curves for the results of the experiment with TF-IDF weighting: (a) logs only; (b) tokens only; (c) logs and tokens simultaneously.

4.2 Results for a proprietary dataset

Fig. 13 shows anomaly graphs for the test anomaly period with different set parameters: with TF-IDF weighting of logs only (Fig. 13a), tokens only (Fig. 13b) and logs and words simultaneously (Fig. 13c). All graphs clearly show the occurrence of anomalies in the middle of the test period on 28 October which led to the failure on 30 October, but other anomalous areas are also highlighted.

Fig. 14 shows the same calculations for another test period with an anomaly that led to a failure of operation on 4 January. In these graphs, anomalies immediately before the failure are recorded only for the model with TF-IDF weighting of logs only (Fig. 14a).

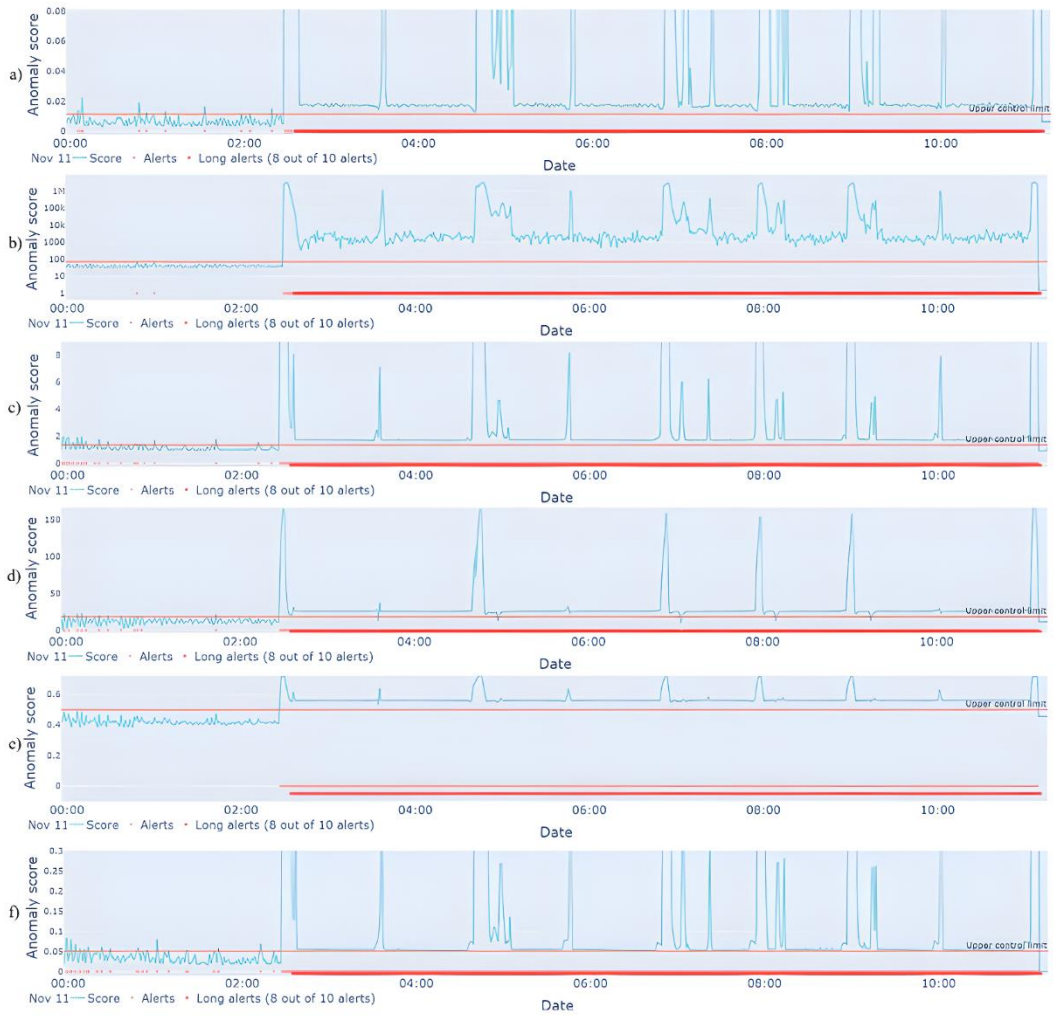

Fig. 12. Anomaly graphs for the test period of the HDFS dataset constructed by the following methods: a) Autoencoder, b) Elliptic Envelope, c) Local Outlier Factor, d) One Class SVM, e) Isolation Forest, f) k-Nearest Neighbours.

5. Conclusions

A key feature of our work is our focus on database logs containing SQL queries. However, open datasets with similar logs labeled specifically for the task of collective anomaly detection are virtually nonexistent. This is confirmed by recent studies [33]: existing log collections lack datasets specific to distributed databases, creating a significant gap for the development and testing of appropriate methods.

Our use of the HDFS dataset was adaptive. These datasets contain labels for point anomalies, and their adaptation to the task of collective detection (by aggregating them into minute blocks) is artificial and imperfect, which, as demonstrated in Section 4.1, could impact metrics (e.g., high FAR). In light of the above, the primary goal of our work was not to outperform all existing methods on standard benchmarks, but to propose and evaluate a comprehensive solution for a new, poorly understood practical problem: anomaly detection in database logs through the analysis of SQL query temporal patterns.

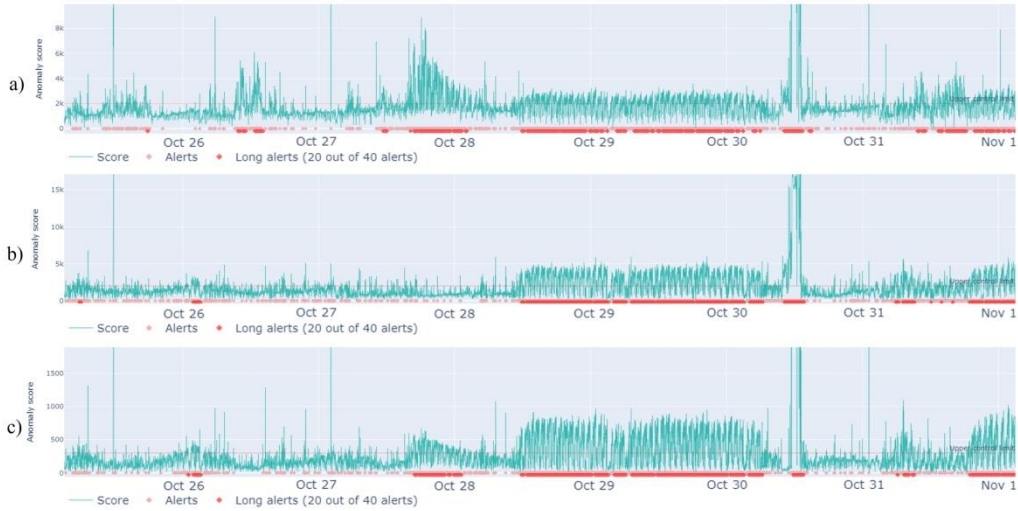

Fig. 13. Anomaly graphs for the 1st test period with TF-IDF weighting: a) logs only; b) tokens only; c) logs and tokens simultaneously.

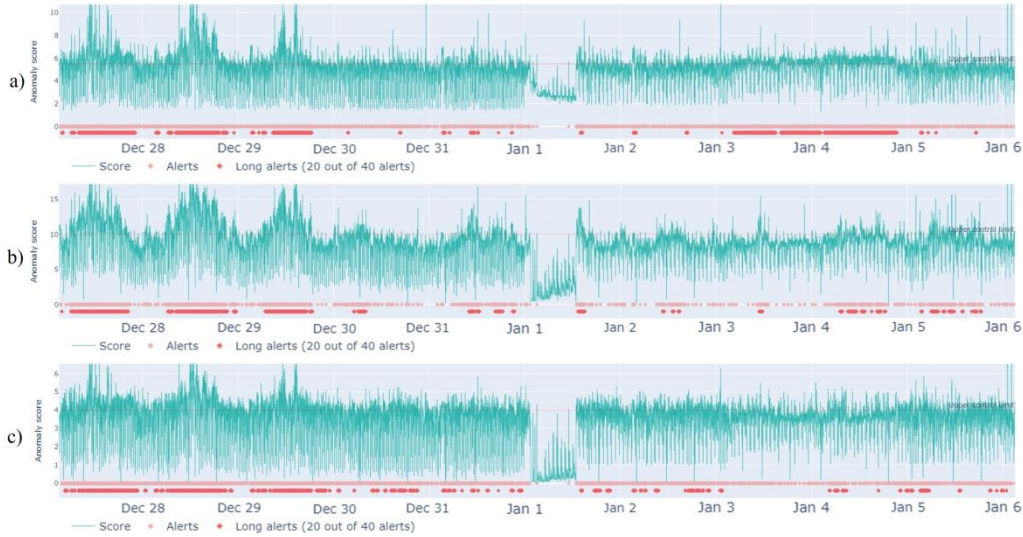

Fig. 14. Anomaly graphs for the 2nd test period with TF-IDF weighting: a) logs only; b) tokens only; c) logs and tokens simultaneously.

Our contribution in this context is as follows:

- Developing a parser that preserves and structures the semantics of SQL queries rather than removes them – a task that standard parsers (Drain, Spell) are not designed to address.
- Proposing a complete pipeline from raw logs to anomaly detection, tailored for working with real Vertica data.
- Proving the effectiveness of the proposed approach on a real Vertica dataset, where the method demonstrated balanced and practical results, in contrast to artifacts obtained on adapted datasets.

Despite the lack of a direct end-to-end comparison in this paper, we believe its value lies in addressing a specific, relevant, and understudied practical problem. We hope that the proposed

solution and discussion of the objective difficulties of comparison will serve as a foundation for future research in this area.

In further research, it is planned to expand the experimental database by collecting additional Vertica database failure incidents, find suitable open labeled datasets, and select the optimal algorithm and its hyperparameters. Future work should also explore robust clustering methods, such as those proposed in [34], to improve anomaly detection reliability under noisy data conditions. In future research, it would be interesting to adapt approaches from works on annotation transfer between languages [35] for automatic anomaly marking in logs. As shown in their work, LLMs are capable of correctly processing structured text data (for example, preserving NER markup during translation), which makes them a promising tool for preprocessing logs, where the accuracy of entity extraction (error codes, process IDs) is critical. For instance, LLMs could generate silver annotations for rare types of anomalies, significantly reducing the cost of manual annotation and expanding the capabilities of automated system monitoring.

References

[1]. Shepherd D. Why DNS exploits continue to be a top attack vector in 2024. Tahawultech, 2024. Available at: https://www.tahawultech.com/insight/why-dns-exploits-continue-to-be-a-top-attack-vector-in-2024, accessed 05.11.2025.

[2]. Shkodyrev V.P., Yagafarov K.I., Bashtovenko V.A., Ilyina E.E. Review of anomaly detection methods in data streams. In Proceedings of the Second Conference on Software Engineering and Information Management, St. Petersburg, Russia, 2017, vol. 1864.

[3]. He P., Zhu J., Zheng Z., Lyu M.R. Drain: An online log parsing approach with fixed depth tree. In IEEE International Conference on Web Services (ICWS), 2017, pp. 33-40. DOI: 10.1109/ICWS.2017.13.

[4]. Du M., Li F. Spell: Streaming parsing of system event logs. In 2016 IEEE 16th International Conference on Data Mining (ICDM), 2016, pp. 859-864. DOI: 10.1109/ICDM.2016.0103.

[5]. Bojanowski P., Grave E., Joulin A., Mikolov T. Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics, 5, 2017, 135-146. DOI: 10.1162/tacl_a_00051.

[6]. Sammut C., Webb G.I. TF-IDF. In Encyclopedia of Machine Learning; Springer US: Boston, MA, 2010, pp. 986-987. DOI: 10.1007/978-0-387-30164-8_832.

[7]. Zhang X., Xu Y., Lin Q., Qiao B., Zhang H., Dang Y., Xie C., Yang X., Cheng Q., Li Z. et al. Robust log-based anomaly detection on unstable log data. In Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), 2019, pp. 807-817. DOI: 10.1145/3338906.3338931.

[8]. Lu S., Wei X., Li Y., Wang L. Detecting anomaly in big data system logs using convolutional neural network. In 2018 IEEE 16th International Conference on Dependable, Autonomic and Secure Computing, 16th International Conference on Pervasive Intelligence and Computing, 4th International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), 2018; pp. 151-158. DOI: 10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00037.

[9]. Du M., Li F., Zheng G., Srikumar V. DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1285-1298. DOI: 10.1145/3133956.3134015.

[10]. Meng W., Liu Y., Zhu Y., Zhang S., Pei D., Liu Y., Chen Y., Zhang R., Tao S., Sun P. et al. LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI), 2019, pp. 4739-4745. DOI: 10.24963/ijcai.2019/658.

[11]. Guo H., Yuan S., Wu X. LogBERT: Log anomaly detection via BERT. In 2021 International Joint Conference on Neural Networks, 2021, pp. 1-8. DOI:10.48550/arXiv.2103.04475.

[12]. Yang L., Chen J., Wang Z., Wang W., Jiang J., Dong X., Zhang W. semi-supervised log-based anomaly detection via probabilistic label estimation. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), 2021, pp. 1448-1460. DOI:10.1109/ICSE43902.2021.00130.

[13]. Han X., Yuan S., Trabelsi M. LogGPT: Log Anomaly Detection via GPT. In 2023 IEEE International Conference on Big Data (BigData), Sorrento, Italy, 2023, pp. 1117-1122. DOI: 10.1109/BigData59044.2023.10386543.

[14]. Guan W., Cao J., Qian S., Gao J. LogLLM: Log-based Anomaly Detection Using Large Language Models, 2024. DOI:10.48550/arXiv.2411.08561.
[15]. Lin Y., Deng H., Li X. FastLogAD: Log anomaly detection with mask-guided pseudo anomaly generation and discrimination. arXiv preprint, 2024. DOI: arXiv:2404.08750.
[16]. Nedelkoski S., Bogatinovski J., Acke A., Cardoso J., Kao O. Self-attentive classification-based anomaly detection in unstructured logs. In 2020 IEEE International Conference on Data Mining, 2020, pp. 1196-1201. DOI: 10.1109/ICDM50108.2020.00148.
[17]. Shchetinin E. Yu., Glushkova A. G., Sevastianov L. A. Anomaly electrocardiograms automatic detection with unsupervised deep learning methods. In Distributed Computer and Communication Networks: Control, Computation, Communications; Vishnevskiy V. M., Samouylov K. E., Kozyrev, D. V., Eds.; Springer Nature Switzerland: Cham, 2022, pp. 117-131. DOI: 10.1007/978-3-031-23207-7_10.
[18]. Xu W., Huang L., Fox A., Patterson D., Jordan M. I. Detecting large-scale system problems by mining console logs. In Proceedings of the 22nd Symposium on Operating Systems Principles, ACM, 2009; pp. 117-132. DOI: 10.1145/1629575.1629587.
[19]. Oliner A. J., Stearley J. What Supercomputers Say: A Study of Five System Logs. In Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2007.
[20]. Kiriachek V., Salpagarov S. Predictive diagnostics of computer systems logs using natural language processing techniques. Discrete and Continuous Models and Applied Computational Science, 2025, 33(2), 172-183. DOI: 10.22363/2658-4670-2025-33-2-172-183.
[21]. GE SmartSignal. Available at: https://www.ge.com/digital/applications/asset-performance-management-apm-software/smartsignal-predictive-analytics, accessed 05.11.2025.
[22]. Farzad A., Gulliver T. A. Unsupervised log message anomaly detection. ICT Express, 2020, pp. 229-237. DOI: 10.1016/j.icte.2020.06.003.
[23]. Catillo M., Pecchia A., Villano U. Autolog: Anomaly detection by deep autoencoding of system logs. Expert Systems with Applications, 191, 2022. DOI: 10.1016/j.eswa.2021.116263.
[24]. Schölkopf B., Platt J.C., Shawe-Taylor J., Smola A.J., Williamson R.C. Estimating the support of a high-dimensional distribution. Neural Computation, 13 (7), 2001, pp. 1443-1471. DOI: 10.1162/089976601750264965.
[25]. Liu F.T., Ting K.M., Zhou Z.H. Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 2008, pp. 413-422. DOI: 10.1109/ICDM.2008.17.
[26]. Breunig M., Kröger P., Ng R., Sander J. LoF: Identifying density-based local outliers. ACM Sigmod Record, 29, 2000, pp. 93-104. DOI: 10.1145/342009.335388.
[27]. Rousseeuw P. J., Van Driessen K. A fast algorithm for the minimum covariance determinant estimator. Technometrics, 41(3), 1999, p. 212. DOI: 10.1080/00401706.1999.10485670.
[28]. Khachumov M.V., Khachumov V.M., Kovalev A.K., Panov A.I. Pattern-Recognition Tools and Their Applications. Pattern Recognit. Image Anal. 2023, 33, 28-38. DOI 10.1134/S1054661823010029.
[29]. Vertica: Big data analytics on-premises, in the cloud, or on hadoop. Available at: https://www.vertica.com/, accessed 03.04.2023.
[30]. Mikolov T., Chen K., Corrado G., Dean J. Efficient estimation of word representations in vector space, 2013. DOI:10.48550/arXiv.1301.3781.
[31]. Pennington J., Socher R., Manning C. D. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1532-1543. DOI: 10.3115/v1/D14-1162.
[32]. Devlin J., Chang M.W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019. DOI: 10.18653/v1/N19-1423.
[33]. Zhang L., Jia T., Jia M., Li Y., Yang Y., Wu Z. Multivariate Log-based Anomaly Detection for Distributed Database, 2024. DOI: 10.48550/arXiv.2406.07976.
[34]. Shibzukhov Z.M. On a Robust Approach to Search for Cluster Centers. Autom Remote Control 82, 1742-1751 (2021). DOI: 10.1134/S0005117921100118.
[35]. Popov D., Terentev E., Serenko D., Sochenkov I., Buyanov I. Transferring Natural Language Datasets Between Languages Using Large Language Models for Modern Decision Support and Sci-Tech Analytical Systems. Big Data Cogn. Comput., 2025, 9, 116. DOI: 10.3390/bdcc9050116.

Информация об авторах / Information about authors

Владислав Анатольевич КИРЯЧЁК – аспирант кафедры математического моделирования и искусственного интеллекта факультета физико-математических и естественных наук РУДН им. Патриса Лумумбы с 2023 года. Сфера научных интересов: обработка естественного языка, большие языковые модели, машинное обучение, нейронные сети, предиктивная диагностика.

Vladislav Anatolyevich KIRIACHEK is a postgraduate student at the Department of Mathematical Modeling and Artificial Intelligence of the Faculty of Physics, Mathematics, and Natural Sciences at Patrice Lumumba Peoples' Friendship University of Russia (RUDN University) since 2023. Research interests: natural language processing (NLP), large language models (LLMs), machine learning (ML), neural networks, and predictive diagnostics.

Солтан Исмаилович САЛПАГАРОВ – кандидат физико-математических наук, доцент кафедры математического моделирования и искусственного интеллекта факультета физико-математических и естественных наук РУДН им. Патриса Лумумбы. Сфера научных интересов: обработка естественного языка, большие языковые модели, машинное обучение.

Soltan Ismailovich SALPAGAROV – Cand. Sci. (Phys.-Math.), an associate professor at the Department of Mathematical Modeling and Artificial Intelligence at the Faculty of Physics, Mathematics, and Natural Sciences at Patrice Lumumba Peoples' Friendship University of Russia. Research interests: natural language processing (NLP), large language models (LLMs), and machine learning (ML).