Duale Hochschule Baden-Württemberg Mannheim

Project Thesis

Semantic Anomaly Detection in Log Messages: A Fusion of Doc2Vec and Isolation Forest

Business Information Systems

Data Science

Author:	Fabian Schlarmann
Matriculation No.:	6458742
Semester:	Second Semester
Company:	Boehringer Ingelheim Pharma GmbH & Co. KG
Department:	High-Performance Computing
Course:	WWI22DSA
Degree Program Director:	Prof. habil. Dr.-Ing. Dennis Pfisterer
Scientific Adviser:	Prof. habil. Dr.-Ing. Dennis Pfisterer
Company Adviser:	Johannes Koppe
Time Frame:	31.07.2023 – 10.11.2023

Statement of Originality

The author of the present work, titled "*Semantic Anomaly Detection in Log Messages: A Fusion of Doc2Vec and Isolation Forest*," has independently written this project thesis and used no other sources than those indicated. The submitted electronic version is in accordance with the printed version.

Biberach, 17.10.2023
Place, Date

Fabian Schlarmann

Permission to Publish

The present project thesis has been granted explicit permission to be published by the company 'Boehringer Ingelheim Pharma GmbH und Co. KG', Binger Straße 173 D-55216 Ingelheim am Rhein.

The contents of the paper may be disclosed in whole or in part. Copies and transcripts - including digital copies - may be made.

Place: Biberach Date: 17.10.23 Signature: J. Koppe Name: J.Koppe

Acknowledgements

I would like to express my gratitude to my company adviser at Boehringer Ingelheim, Johannes Koppe, for his invaluable guidance and support throughout the course of this project thesis. His expertise and insights have been instrumental in shaping my understanding of this topic.

Furthermore, I would like to thank Dr. Moritz Schneider, a Senior Data Scientist at Boehringer Ingelheim, for providing useful hints and assistance regarding various aspects of my work. His expertise and guidance have been invaluable in helping me navigate through the challenges of this project.

Additionally, I would like to extend my appreciation to my fellow student colleagues at Boehringer Ingelheim, Enzo Schmitz and Tobias Welti, who likewise worked on their project theses, similar to mine. Their project theses are titled "Analysis of Batch Job Accounting Data in the High-Performance Computing Cluster" and "Leveraging Artificial Intelligence for Process Improvement: A Comprehensive Method to Minimize Manual Effort in Incident Ticket Data Analysis through Efficient Trend Detection". Our collaborative discussions and exchange of ideas have significantly enriched my research experience and contributed to the success of this project thesis.

Last but not least, I would like to express my gratitude to my scientific advisor from university, Prof. Dr. Dennis Pfisterer, who has been of great help with formal issues. His guidance and support have been essential in ensuring the proper completion of this project thesis.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

BI	Boehringer Ingelheim
HPC	High-Performance Computing
DHBW	Duale Hochschule Baden-Württemberg

Abstract

This project thesis introduces a machine-learning method designed to automatically detect semantic anomalies in log messages, thereby enhancing system security and reliability. The method involves transforming log messages into semantically-driven vectors and then applying anomaly detection techniques to them.

Using Doc2Vec and the Isolation Forest algorithm, this specific approach complements human analysis, laying the groundwork for future semantic anomaly detection research. Performance is assessed using metrics such as accuracy, precision, recall, and F1-Score.

After implementation in Boehringer Ingelheim's High-Performance Computing (HPC) systems, the method effectively identifies semantic anomalies in an initial attempt.

1 Introduction

1.1 Machine Learning for Anomaly Detection in Log Messages

Machine learning offers significant potential for detecting anomalies in log messages. This can enhance cybersecurity, system monitoring, and fraud detection. [6] Generally, anomalies in log messages can be categorized into two groups [2]:

1. **Semantic anomalies**: These log messages significantly deviate from the usual content and are typically rare in terms of the written information they contain.
2. **Numerical anomalies**: These log messages behave abnormally in terms of their frequency or other numerical characteristics within a certain time slot.

By leveraging machine learning techniques, these anomalies can be effectively identified, thereby enhancing the security and performance of systems.

This project thesis is particularly focused on detecting semantic anomalies.

The Relationship between Log Data and Semantic Anomaly Detection

General log data, consisting of time-stamped log records composed of two parts the 'Log Header' and the 'Log Message', provides a detailed description of events, activities, or communications from software programs, systems or devices. These records serve various purposes, such as monitoring system performance, troubleshooting system functionality issues and investigating security incidents. [1]

The importance of log data is intrinsically linked to the identification of anomalies, as can be derived from the following excerpt:

> "System logs require careful monitoring, because each anomaly in the system log may have a potential impact on service availability, data confidentiality, or integrity. However, large-scale systems generate such a large number of logs that human supervision of them becomes both very difficult and expensive. What is more, security specialists may sometimes overlook a suspicious entry among thousands of others." [36, p.1]

Therefore, recognizing atypical semantic patterns as well as numerical ones automatically within the log messages of general log data enables IT specialists to analyze log data more efficiently. This is particularly true when dealing with reduced log data subsets composed predominantly of semantic anomalies.

1.2 Objective and Motivation of this Project Thesis

Currently, at Boehringer Ingelheim (BI), the main way of analyzing log data is by manual inspection, mostly done in a reactive manner. This often results in the identification of system issues only after a crash or abnormal behavior has occurred, prompting the High-Performance Computing (HPC) team to examine the log files.

As a result, this project thesis primarily aims to answer the following research question:
Which approach is suited for automatically detecting semantic anomalies in log messages?

To address this question, the main objective is to develop a general machine-learning-based methodology for detecting semantic anomalies automatically within log messages, ultimately contributing to the establishment of more secure and reliable systems. Therefore, the emphasis is placed on comparing various measurements derived from a confusion matrix, such as precision, recall, and F1-Score, to assess the quality of the detection process.

1.3 Contribution

The proposed methodology introduces the combination of vectorization and anomaly detection techniques. By working in tandem, these complementary techniques efficiently identify atypical semantic patterns in log messages, complementing manual human analysis.

First, the log messages are extracted from the log entries. Following this, a composite model, in a specific illustration a synergistic combination of Doc2Vec and the Isolation Forest algorithm, is trained using log messages that are representative of the behavior within a certain system, along with the optimal default values for the parameters that need to be determined for each distinct synergistic combination individually. Nevertheless, general optimal default values for the parameters that tend to yield the best results with the datasets and the specific composite model used here are provided.

In general, these composite models function in a way that they translate any log messages into a high-dimensional vector space according to their respective underlying semantics. These vectors are then learned by an anomaly detection technique, which leads to the ability to detect unusual semantic patterns.

2 Background

2.1 Boehringer Ingelheim (BI)

Boehringer Ingelheim, a global, research-driven pharmaceutical company with a rich history spanning over 130 years, specializes in providing innovative treatments in both human and veterinary medicine, committing to enhancing health and quality of life. As one of the world's leading pharmaceutical companies, it maintains a robust presence in both developed and emerging markets. Furthermore, Boehringer Ingelheim's primary areas of interest encompass respiratory diseases, metabolism, immunology, oncology, and diseases of the central nervous system. [19] Recognizing the importance of technological advancements, the company actively partners with IT Innovation Drivers such as Google, emphasizing the crucial role that Information Technology, particularly Artificial Intelligence and Quantum Computing, already plays in shaping the future of healthcare and drug development. [28] [21]

2.2 High-Performance Computing (HPC) at BI

High-Performance Computing is a vital tool for pharmaceutical companies like Boehringer Ingelheim (BI), as it accelerates drug discovery by rapidly processing large datasets and complex simulations improving the accuracy of drug interaction predictions [37, p.4f], which in turn reduces development costs and time-to-market. [20]

Additionally, as correctly noticed by researchers, High-Performance Computing enables the advancement of personalized medicine by permitting an extensive analysis of both genetic and clinical data [37, p.4f], thereby leading to the development of more effective treatment strategies. Therefore, HPC is essential for maintaining competitiveness and driving progress in the pharmaceutical industry. [18]

Given the complexity and diversity of tasks managed by (HPC) systems, they often require simultaneous monitoring and maintenance of multiple systems. And, as these systems, in their operation, generate a substantial amount of log data, this data is a crucial resource for identifying potential system issues. [1]

2.3 Log Entries within the HPC environment

This log data, being directly derived from these systems, offers a comprehensive record of system activities and operations. [1] Generally, within the HPC environment, log entries show the following characteristics:

```
Jul 30 00:08:24 inhcc[redacted] systemd-logind: Removed session 720409.
Jul 30 00:08:24 inhcc[redacted] systemd: Removed slice User Slice of x2[redacted].
Jul 30 00:08:25 inhcc[redacted] telegraf: 2023-07-29T22:08:25Z E! [outputs.influxdb] When writing to [https://bi-hpc[redacted]]: failed doing req: Post "https://bi-hpc[redacted]": dial tcp [redacted]: connect: connection refused
Jul 30 00:08:48 inhcc[redacted] systemd: Created slice User Slice of x2[redacted].
Jul 30 00:08:48 inhcc[redacted] systemd-logind: New session 720410 of user x2[redacted].
Jul 30 00:08:48 inhcc[redacted] systemd: Started Session 720410 of user x2[redacted].
Jul 30 00:08:51 inhcc[redacted] systemd-logind: Removed session 720410.
Jul 30 00:08:51 inhcc[redacted] systemd: Removed slice User Slice of x2[redacted].
Jul 30 00:09:04 inhcc[redacted] be[AD1]: Could not start TLS encryption. unknown error
Jul 30 00:10:02 inhcc[redacted] systemd: Started Session 720421 of user root.
```

Figure 2.1: Examples of Log Entries within the HPC environment

Most log entries, when retrieved from Linux systems, follow the format shown above; if not, they can be adjusted to fit that format. Based on this, the structure of these log entries can be determined as follows:

Log Header			**Log Message**
Jul 30 00:10:02 Timestamp	inhcc[redacted] Hostname	systemd Servicename	Started Session 720421 of user root.

Table 2.1: Log Entries within HPC environment

2.4 Definition "Semantic Anomalies"

A clear definition of the general term 'Anomaly' is provided in this paper:

> "Anomalies are occurrences in a dataset that are in some way unusual and do not fit the general patterns." [10, p.297]

Here is an example of what a semantic anomaly might look like in the HPC environment:

```
Jul 30 00:09:04 inhcc[redacted] be[AD1]: Could not start TLS encryption. unknown error
```

Here is an example of what a semantic anomaly would **not** look like in the HPC environment:

```
Jul 42 42:42:42 inhcc[redacted] systemd-logind: New session 424242 of user 'Hi_RareUser_Hi'.
```

It is just a user that logs in quite rarely.

Based on these examples, a semantic anomaly in the given context refers not only to a log entry displaying unusual patterns, which are not typically observed when systems function properly, but particularly to those entries that deviate significantly from the norm in terms of content. It is important to note that the classification of a log message as a semantic anomaly is not solely based on its frequency of occurrence. For this reason, it is essential to consider the underlying semantics of a log message.

As demonstrated later in this project thesis, the log messages are translated into a multidimensional space using vectorization techniques, which assign a vector to each log message. Consequently, the semantic anomalies detected in this multidimensional space can be referred to as 'Multidimensional numerical anomalies' as well. [10, p.308]

2.5 Machine Learning

A clear definition of the general term 'Machine Learning' is provided in this paper:

> "Machine learning describes the capacity of systems to learn from problem-specific training data to automate the process of analytical model building and solve associated tasks." [22, p.685]

Within this project thesis, the capacity of 'Machine Learning' is then used to detect unusual semantic patterns.

2.6 Supervised Learning and Unsupervised Learning

Furthermore, in another paper, they state:

> "Machine learning can be classified as either supervised or unsupervised. Supervised algorithms can apply past knowledge to new data whereas unsupervised algorithms make conclusions from datasets." [17, p.1]

In the project thesis discussed here, the model components used to detect anomalous log messages within extensive log files are categorized as unsupervised methods. This classification is derived from their ability to learn semantic patterns in log messages without the need for provided labels, subsequently allowing them to classify previously unseen log messages without labels as well, based on the patterns acquired from historical training data. [17, p.1]

2.7 Evaluation Metrics

In the approach presented here, a composite model with a multitude of adjustable parameters is trained. As each specific implementation results in varying outcomes, some prove to be more effective than others. To determine the efficiency of specific implementations among those tested, the approach depends on specific metrics. By assessing the performance of each tested specific implementation, these metrics ultimately pinpoint the one that leads to the best results. [42] Therefore, the performance of different specific implementations is measured by the following four evaluation metrics:

2.7.1 Accuracy

Accuracy is a general performance measure that represents the proportion of correctly predicted observations out of the total observations. A high accuracy indicates that the composite model performs well. [42] Mathematically, accuracy can be defined as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{2.1}$$

where TP denotes True-Positives, TN denotes True-Negatives, FP denotes False-Positives, and FN denotes False-Negatives.

However, it is not recommended to rely solely on this metric because, in the presence of class imbalance when dealing with anomalies, accuracy may become an unreliable measure of performance. Therefore, it is necessary to consider class-specific performance metrics as well. The metrics that fulfill this need more precisely are Precision, Recall, and F1-Score. [42]

2.7.2 Precision

In the context of this project thesis, Precision is a performance measure that represents the proportion of correctly identified anomalies out of the total anomalies detected by the model. A high precision indicates that the composite model is reliable in its detection of anomalies. [42] Mathematically, precision can be defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.2}$$

where TP denotes True-Positives and FP denotes False-Positives.

2.7.3 Recall

Within the scope of this project thesis, Recall is a performance measure that refers to the proportion of real anomalies detected by the model out of all the existing anomalies in the dataset. A high recall indicates that the composite model is effective at identifying anomalies within the data. [42] Mathematically, recall can be defined as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.3}$$

where TP denotes True-Positives and FN denotes False-Negatives.

2.7.4 F1-Score

Pertaining to this project thesis, the F1-score is a performance measure that combines both precision and recall, the harmonic mean, to provide a single metric for evaluating the effectiveness of a model in detecting anomalies. A high F1-Score indicates that the composite model is effective at identifying anomalies within the data while maintaining a low rate of false alarms. [42] Mathematically, the F1-Score can be defined as:

$$\text{F1-Score} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})} \tag{2.4}$$

2.8 Doc2Vec

Doc2Vec is an unsupervised learning algorithm. It has been shown to be more precise at generating appropriate embeddings from a general perspective compared to other approaches. [24, p.4] It is an extension of **Word2Vec**, a similar approach, which is used to create vector representations of words in a high-dimensional space. [24, p.1]

While Word2Vec generates vector representations for individual words, Doc2Vec expands upon this idea by creating vector representations for entire sentences, paragraphs, or even whole documents, introducing a special 'Paragraph Vector'. By doing so, the Doc2Vec is able to effectively capture the context of words within a document, as well as semantic and syntactic similarities, meaningful relationships with other words, and various linguistic aspects. [25, p.3]

In order to achieve this, Doc2Vec is trained to predict a word given its context (a set of surrounding words) in a document, or vice versa. It does this by learning to maximize the probability of a word given its context in the document. As a result, the document vectors can then be used for various tasks such as document similarity, document clustering, or as input for further downstream tasks like classification, regression, or information retrieval. [25, p.8]

The parameters that seem to have the most influence when training a Doc2Vec model are as follows [14] [24, p.5]:

- **dbow_words**: The dbow_words parameter determines whether the DBOW model should train word vectors alongside document vectors (when set to 1) or only train document vectors (when set to 0).
- **vector size**: The size of the vector representation for documents and words. A larger vector size can capture more information but may require more computational resources.
- **window**: The number of context words considered around a target word. A larger window size captures more context but may increase training time.
- **min_count**: The minimum frequency of words to be included in the vocabulary. Words with a frequency lower than this threshold will be ignored during training.
- **negative**: The number of negative samples used during training. This parameter affects the noise-contrastive estimation, which helps improve training efficiency.
- **epochs**: The number of iterations over the entire dataset during training. More epochs can lead to better performance but may also increase the risk of overfitting.

2.9 Isolation Forest

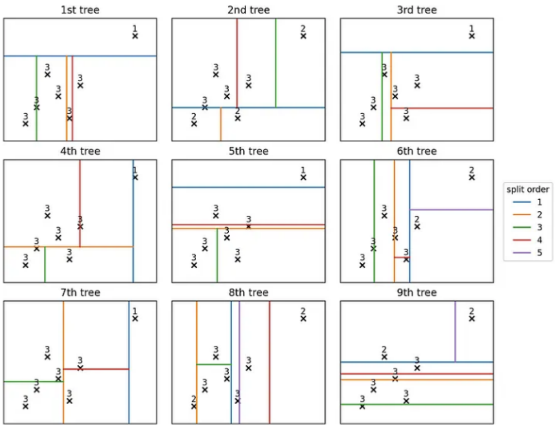

Figure 2.2: Two-dimensional illustration of various trees that make up an Isolation Forest [27]

The Isolation Forest is an unsupervised learning algorithm primarily used for anomaly detection, operating on the principle that anomalies are data points that are few and different, and hence, can be isolated more readily. By constructing multiple decision trees, known as isolation trees, using a subset of the training data, the algorithm recursively partitions the data based on randomly selected features and split values until each data point is isolated or a specified tree depth is reached. [27]

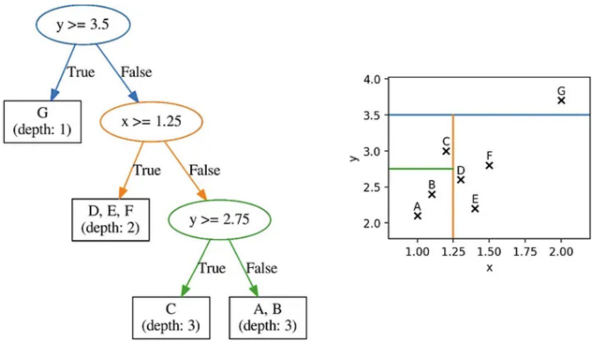

Figure 2.3: Two different visual representations of the same randomly fit tree [27]

Following this, a path length, defined as the number of conditions used to isolate a data point, is calculated for each data point across all trees, and the average path length for each data point is computed, serving as the basis for the anomaly score. Notably, a shorter average path length suggests that the data point is more likely an anomaly, as it can be isolated more quickly. [43, p.6] To classify anomalies, a **percentage threshold**, representing the expected proportion of anomalies in the training dataset, is set, and data points with the highest anomaly scores, up to this threshold, are flagged as anomalies. For anomaly detection in new data, the algorithm calculates the average path length within the "forest of trees" built when training - as can be seen in figure 2.3 - and derives the anomaly score for each test data point, flagging it as an anomaly if the score exceeds the percentage threshold. [39]

Overall, the Isolation Forest algorithm is particularly effective for high-dimensional datasets [43, p.1], and offers efficient anomaly detection with minimal parameter tuning. [39] As this aligns perfectly with the fact that the log messages will be translated into a high-dimensional vector space, the Isolation Forest algorithm is well-suited for extracting semantic anomalies.

3 Related Work

The company adviser provided the essential environment, data, and other resources necessary for implementing the anomaly detection approach for log messages introduced in this project thesis. In this chapter, a brief literature review is presented to offer a deeper understanding of previous methodologies used to address similar questions. This approach will facilitate discussions to more easily identify strategies that effectively fulfill the purpose of this research.

In the 2022 paper [45] "A Docker Container Anomaly Monitoring System Based on Optimized Isolation Forest", a dual-method approach for detecting and analyzing anomalies in container environments is proposed. This approach relies on the Isolation Forest algorithm which is utilized to analyze the container's resource metrics, detecting anomalies that could indicate potential issues. When anomalous behavior is identified, the system initiates deterministic log analysis of the log entries within the detected anomalous container to determine the cause. This analysis process encompasses preprocessing, mining frequent itemsets, and rule comparison, ultimately leading to the identification of specific events or patterns associated with the detected anomaly.

In the 2016 paper [41] "Detecting Anomalous User Behavior Using an Extended Isolation Forest Algorithm: An Enterprise Case Study" the authors introduce a framework for detecting anomalous user behavior. This framework is built upon an 'extended version' of the Isolation Forest algorithm, which is designed to process log data, including especially categorical data. The algorithm is fed with features extracted directly from the log data, enabling it to effectively identify anomalous user behavior. A key insight from their work is the significant influence of login time on the perception of anomalous behavior. [41, p.12f].

In the 2018 master's thesis [15], "Anomaly Detection for Application Log Data", the key finding is that an

> "[...] encoder learns a vector representation of the input features, and the decoder uses this to reconstruct the time-series." [15, p.35f]

approach performs better compared to three different other approaches that rely solely on single methods without being complemented by a second one. The single methods used were K-Means Clustering, K-NN Global Density-based, and LSTM Neural Network.

In the highly influential 2016 paper [16] recognized by the International Symposium on Software Reliability Engineering as one of the most significant contributions to the community in the past 30 years [30], the authors introduce state-of-the-art automated anomaly detection approaches. Their comprehensive overview covers six advanced log-based anomaly detection techniques, specifically applied to two production log datasets, HDFS and BGL, they obtained after reaching out to their respective scientific authors [16, p.212]. Through a rigorous assessment of accuracy and efficiency, the authors skillfully compare three supervised and three unsupervised approaches in various settings [16, p.213f]. Their key findings demonstrate that the combination of vectorization techniques with anomaly detection techniques is able to significantly outperform manual human analysis, highlighting the value of automated approaches in the field of anomaly detection. [16, p.216]

In 2020 an interesting paper has been introduced mentioning [9] that

> "[...] an unsupervised model for log message anomaly detection is proposed which employs Isolation Forest and two deep Autoencoder networks. The Autoencoder networks are used for training and feature extraction, and then for anomaly detection, while Isolation Forest is used for positive sample prediction. The proposed model is evaluated using the BGL, Openstack and Thunderbird log message data sets. The results obtained show that the number of negative samples predicted to be positive is low, especially with Isolation Forest and one Autoencoder. Further, the results are better than with other well-known models." [9, p.1]

In 2022 a paper [40] proposed

> "a log anomaly detection method, namely Prog-BERT-LSTM, which uses the network based on the BERT model as the text vectorization module, and designs the sequence feature learning module based on LSTM" [40, p.1]

In 2020, 'Loghub', a project [44] is published offering open-source, partially labeled log datasets to promote AI-driven log analytics research. With 19 real-world datasets from diverse software systems, 'Loghub' has been downloaded over 90,000 times. The authors discovered that 35% of log data is primarily used for anomaly detection, underlining the significance of effective log analysis techniques. [44, p.5] To address challenges like the need for diverse log datasets and the performance of anomaly detection approaches, they conducted experiments using six labeled log datasets from 'Loghub'. The results revealed that supervised approaches outperformed unsupervised ones, but the scarcity of labeled data in real-world scenarios highlighted the necessity for more advanced log data analysis methods, especially those dealing with unlabeled log data. [44, p.9]

4 Methodology

In this research, two methods were employed to address the research question: a content-adressed discussion of 'Related Work' and conducting an 'Experiment'. Since the objective of this project thesis is to pinpoint an effective technique for automatically detecting semantic anomalies within log messages, firstly previous studies are discussed on machine learning methods that have been applied to similar challenges. This leads to a comprehensive strategy that guides the selection of a suitable approach for the development of composite models for semantic anomaly deetection. After reviewing related work on existing machine learning algorithms in the 'Log Messages' domain, an approach that is grounded in robust assumptions can be established.

4.1 Content-adressed Discussion of Related Work

- Regarding this paper [45], this approach has certain limitations. One potential weakness is its inability to detect anomalies that do not significantly impact the container's resource metrics. Since the Isolation Forest algorithm in this case primarily focuses on numerical data derived from these metrics, it might be less sensitive to issues that do not cause noticeable changes in these values but are hidden directly in the log messages. This could potentially result in no log entries being extracted from a container that is actually behaving anomalously, but in a different way. Furthermore, as the authors themselves state:

  > "[...], if none of the matches is successful, the administrator selects the frequent itemsets and adds them to the normal rule database and the exception rule database." [45, p.7]

  Based on this statement, it is evident that another limitation of their approach is the need for manual adjustments to the log analysis at certain times, as it relies on deterministic methods. This aspect could be improved by examining it from a different perspective: By focusing on learning the general meaning of log messages, they can be converted into semantically meaningful vectors. From these vectors, it can then be automatically determined if the log messages represent anomalous behavior. This process eliminates the need for extracting log messages only from specific containers and making manual adjustments, as it could be applied universally to all log messages generated by each container.

- In the paper [41], the authors feed their 'extended version' of the Isolation Forest algorithm with vectors that are based directly on numerical or categorical features retrieved from the log data itself. These features are extracted in a way that makes it possible to obtain single features from the vectors, which, when standing alone, represent useful information and can be seen as real-world features. In contrast, from a semantically-driven point of view it seems to be more reasonable vectorizing log messages, complete sentences, in a way that the vectors semantically represent the log messages. In this case, the vectors are composed of features where each one alone does not represent any useful value; only as a whole do they represent the log message.

- Similar to the proposed approach, the master's thesis [15] introduces a composite method as well. However, this approach has two weaknesses:

  The first weakness is that his focus is on learning to reconstruct the time-series data, which requires clean normal instances for training, as he states:

  > "The intuition here is that the encoder-decoder pair would only have seen normal instances during training and learned to reconstruct them." [15, p.36]

  In real-world scenarios, it is often challenging to train algorithms with clean data composed only of normal instances. Therefore, in this specific implementation, the composite model is trained using log data directly retrieved from real-time working systems to maintain a realistic environment.

  The second weakness lies in the technique used to create the vectors for the log entries, as his approach focuses on the frequency of log entries without capturing the underlying semantic meaning of them [15, p.28]. Although the frequency should be taken into account as well, unlike techniques such as Doc2Vec, his approach does not inherently learn semantically-driven representations, which can lead to difficulties in detecting anomalies that might appear more frequently, as they could be positioned closer to normal instances in the vector space. By considering the semantic meaning of these anomalies, the assumption is that they are less likely to be overlooked, as they would be more likely assigned to vectors not located near vectors representing normal instances.

- While their work [16] strongly emphasizes the capabilities of anomaly detection techniques when already provided with vector input representing log data [16, p.208], it does not precisely highlight the fact that this approach essentially consists of two distinct techniques: one for vectorizing data and another for detecting anomalies within that data. Their main focus lies in explaining that, when already having vector input, they can be used for training anomaly detection techniques. However, they mention the important vectorization step, which shares the same assumed weakness as in the master thesis, as both scientific works translate log data in the same way using a so-called 'event count vector' [15, p.28] [16, p.208], **only as a side note**, without providing a comprehensive overview or suggesting that other vectorization techniques could have been used in this process.

As a complement to their previous work [16], this project thesis primarily focuses on clearly illustrating the general approach of how these composite models function, especially in the context of log messages and a particular focus on unsupervised methods that have greater practical applications in the industry when addressing real-world problems with limited labeled log data. Since numerous techniques already exist for generating vectors from sentences or log messages as well as for identifying anomalies in numerical data, the unique contribution of this project thesis lies in the generalization of such composite models.

- In addition to that, in the papers [9] and [40], the authors present more detailed examples of how vectorization techniques, specifically Autoencoders and BERT, along with anomaly detection techniques such as the Isolation Forest algorithm and a type of recurrent neural network, can effectively complement each other. These combinations establish a robust foundation for automatically detecting semantic anomalies in log data, particularly as they are based on the same open-source project 'Loghub' [44].
- 'Loghub' [44] offers valuable data; however, it lacks labeled log data for operating systems such as Windows, Linux, or Mac OS Log, which would have been essential for this research as it focuses on Linux systems. Consequently, it was necessary to collaborate with domain experts at BI to successfully label internal log data for the analysis.

It's worth noting that when there are substantial amounts of legacy log data available, and the necessary resources are at hand, it appears beneficial to train general models with this legacy log data. Relying on models trained with general legacy log data are able to develop the ability to generalize across new log data sets, especially log messages. [8, p.215]

> "This is done by assuming there are semantic similarities in different systems' log data." [8, p.209]

4.1.1 Conclusion of Content-addressed Discussion

Taking into account the related work discussed in this context, it appears to support the assumption that such complementary composite models demonstrate significant potential for automatically detecting semantic anomalies in log messages. As a result, it is reasonable to assume that a general methodology for composite models can be derived. This approach can then be further validated by a distinct implementation of mine, which has not yet been mentioned in the existing literature.

4.2 Experiment

This section is divided into four main components: 'Experiment Setup', 'Data Preprocessing', 'General Approach', and 'Specific Implementation'. The 'Experiment Setup' offers an overview of the experimental environment. 'Data Preprocessing' discusses the details of parsing, particularly using certain Python libraries. Additionally, the 'General Approach' describes the general approach derived from ideas within the related literature review. Finally, the 'Specific Implementation' section presents a practical application of the general approach, combining Doc2Vec and the Isolation Forest algorithm. This specific implementation, after being applied to expert-labeled test data, is subsequently evaluated using metrics derived from a confusion matrix.

4.2.1 Experiment Setup

Environment

In this project thesis, Python was utilized as the primary programming language, a decision influenced by its accessibility and resemblance to natural language due to its simplified syntax. This choice was further complemented by the use of a variety of Python and machine learning libraries, which proved to be essential for the successful execution of the algorithms. [3]

Furthermore, in terms of the origin of the data for the experiment, the log message extraction was conducted in the following environment: Firstly, the log messages were generated by various system processes, services, and applications, and were collected using the 'rsyslog' daemon,

Linux Distribution	Syslog Daemon	Log Storage Directory
Scientific Linux	rsyslog	/var/log

Table 4.1: System Configuration

> "[...] an open-source software utility used on UNIX and Unix-like computer systems for forwarding log messages in an IP network.". [4]

Secondly, the log messages were stored in the '/var/log' directory, with each log source having its own designated file or set of files.

Important Python Libraries Used

Gensim:

> "Gensim is a free open-source Python library for representing documents as semantic vectors, as efficiently (computer-wise) and painlessly (human-wise) as possible. Gensim is designed to process raw, unstructured digital texts ("plain text") using unsupervised machine learning algorithms. The algorithms in Gensim [...] automatically discover the semantic structure of documents by examining statistical co-occurrence patterns within a corpus of training documents. These algorithms are unsupervised, which means no human input is necessary – you only need a corpus of plain text documents." [13]

NumPy:

> "NumPy is an open source project that enables numerical computing with Python. It was created in 2005 building on the early work of the Numeric and Numarray libraries. NumPy will always be 100% open source software and free for all to use. It is released under the liberal terms of the modified BSD license." [31]

Pandas:

Pandas is a Python library designed for efficient data manipulation and analysis, which offers a versatile DataFrame object for seamless data handling. By providing tools for reading and writing data in various formats, it enables intelligent data alignment. The library supports flexible reshaping, pivoting of datasets, high-performance merging, and joining, all while utilizing hierarchical axis indexing. Additionally, Pandas boasts time series functionality and is optimized for performance through Cython integration. Its widespread use spans fields such as finance, neuroscience, economics, and web analytics. [33]

RegEx:

> "A RegEx is a powerful tool for matching text, based on a pre-defined pattern. It can detect the presence or absence of a text by matching it with a particular pattern, and also can split a pattern into one or more sub-patterns. The Python standard library provides a re module for regular expressions. Its primary function is to offer a search, where it takes a regular expression and a string. Here, it either returns the first match or else none." [12]

Scikit:

> "Scikit-learn is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection, model evaluation, and many other utilities." [38]

4.2.2 Data Preprocessing

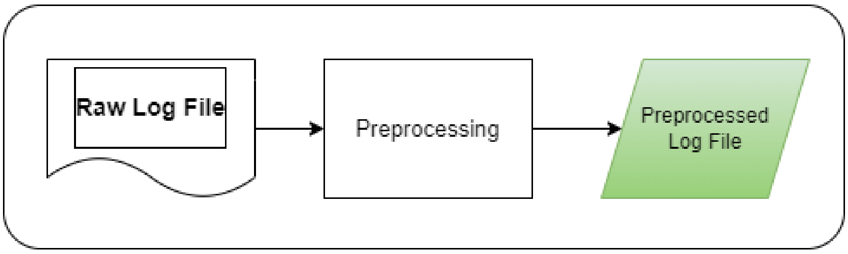

Figure 4.1: Overview Data Preprocessing

Data preprocessing is used to extract log messages from raw log files and convert them into a structured and cleaned format that is easier to analyze. Additionally, the ability to associate log messages with their corresponding log headers is preserved.

4.2.3 General Approach

Model Generation

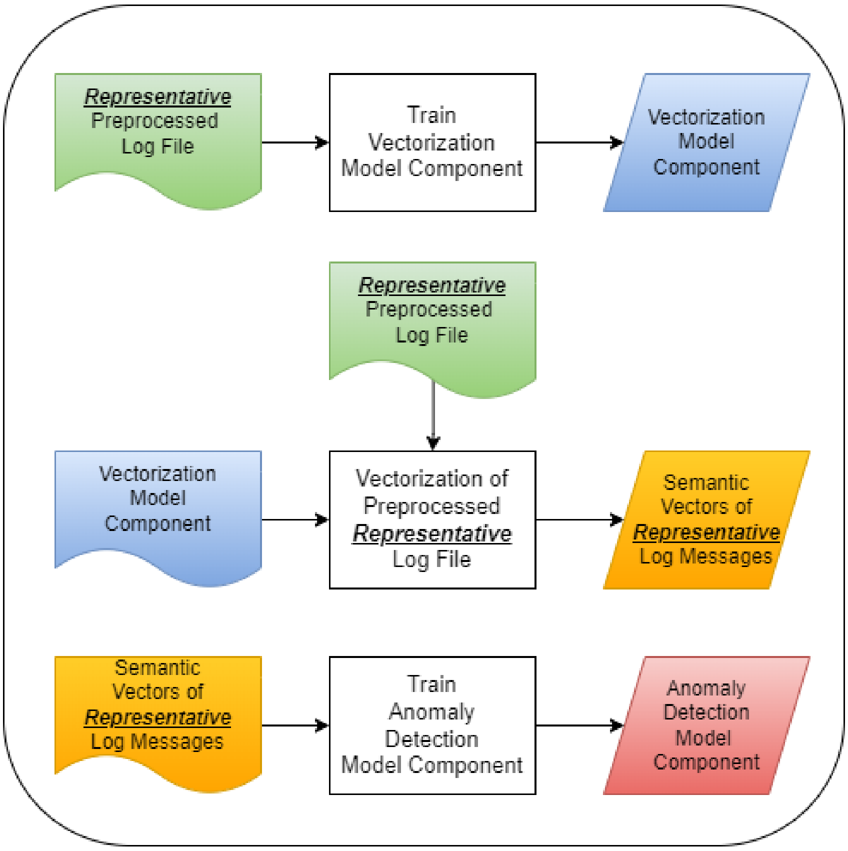
Representative Preprocessed Log File
Train Vectorization Model Component
Vectorization Model Component
Representative Preprocessed Log File
Vectorization Model Component
Vectorization of Preprocessed Representative Log File
Semantic Vectors of Representative Log Messages
Semantic Vectors of Representative Log Messages
Train Anomaly Detection Model Component
Anomaly Detection Model Component

Figure 4.2: Overview Composite Model Generation

Algorithm 1 Composite Model Generation: Vectorization & Anomaly Detection

Let M_1 represent the *vectorization* model component and M_2 represent the *anomaly detection* model component.
Given a dataset D (training data), a composite model is generated as follows:
1. Train the *vectorization* model component M_1 using the dataset D and its respective parameters:
$M_1 = f_1(D, other\_params)$
2. Generate input vectors for the *anomaly detection* model component M_2 using the trained *vectorization* model M_1:
$V = M_1(D)$
3. Train the *anomaly detection* model M_2 using the input vectors V and its respective parameters:
$M_2 = f_2(V, other\_params)$
where f_1 and f_2 are the functions that generate the respective models from the dataset D with the specified parameters.

Both the graphic and the formal description showcase a three-stage process for developing a semantically-driven anomaly detection system. This process entails training a vectorization model component to transform log messages into relevant semantic vectors, which are then employed to train an anomaly detection model component, ultimately enhancing the system's ability to accurately identify semantic anomalies that are of interest.

Anomaly Detection

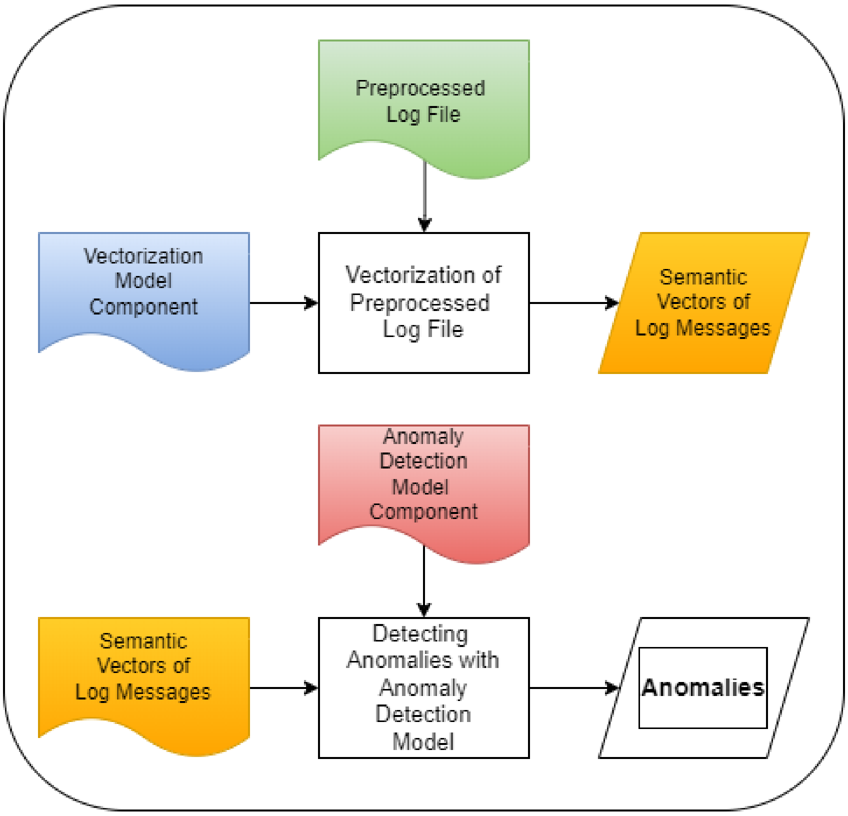
Preprocessed Log File
Vectorization Model Component
Vectorization of Preprocessed Log File
Semantic Vectors of Log Messages
Anomaly Detection Model Component
Semantic Vectors of Log Messages
Detecting Anomalies with Anomaly Detection Model
Anomalies

Figure 4.3: Overview Anomaly Detection

After training, any log messages from a specific system can be used as input for any corresponding trained composite model to detect semantic anomalies.

4.2.4 Specific Implementation

A Fusion of Doc2Vec and Isolation Forest

In this specific implementation, Doc2Vec is employed for vectorization and the Isolation Forest algorithm is employed for semantic anomaly detection. These two model components function synergistically to form a composite model and cannot operate independently. For a representative dataset that mirrors the behavior of the previously described technical environment, a distinct combination of these model components is generated, underscoring their interdependence:

Parameters	Default Values
dbow_words	0
vector_size	300
window	15
min_count	5
negative	5
epochs	20
n_estimators	100
percentage_threshold	1,2,3,4,5,6,7,8,9 and 10 (freely adjustable after training the composite model)

Table 4.2: Default Values for the Parameters of Specific Composite Model

The optimal default values for the parameters for both model components as a whole are essential for the effective generation of a composite model. As such, these default values for the parameters of the model components were obtained from relevant literature [24] [39], which served as a solid base for the training of this specific composite model.

Evaluation

To guarantee dependable conclusions in the context of this specific implementation and to showcase the overall effectiveness of the general approach, the default values for these parameters were consistently applied across 1,000 training iterations , resulting in 1,000 models, 100 models for each value of the 'percentage_threshold' parameter within the interval [1, 10]. General information regarding the datasets used is provided here:

Data Attributes	Train	Test
Labels Available	No - Unlabeled Train Data	Yes - Expert-Labeled Test Data (Labels Only for Subsequent Comparison)
File Size	1.5 GB (aprox. 20×10^6 Log Entries)	8 KB (100 Log Entries)
Origin of Data	Retrieved from Real-Working Systems	Artificially Created (But Based on the same Real-Working Systems)

Table 4.3: Attributes Data

This consistent methodology enabled a precise evaluation of critical performance indicators, including recall, precision, F1-score, and accuracy. The results are as follows:

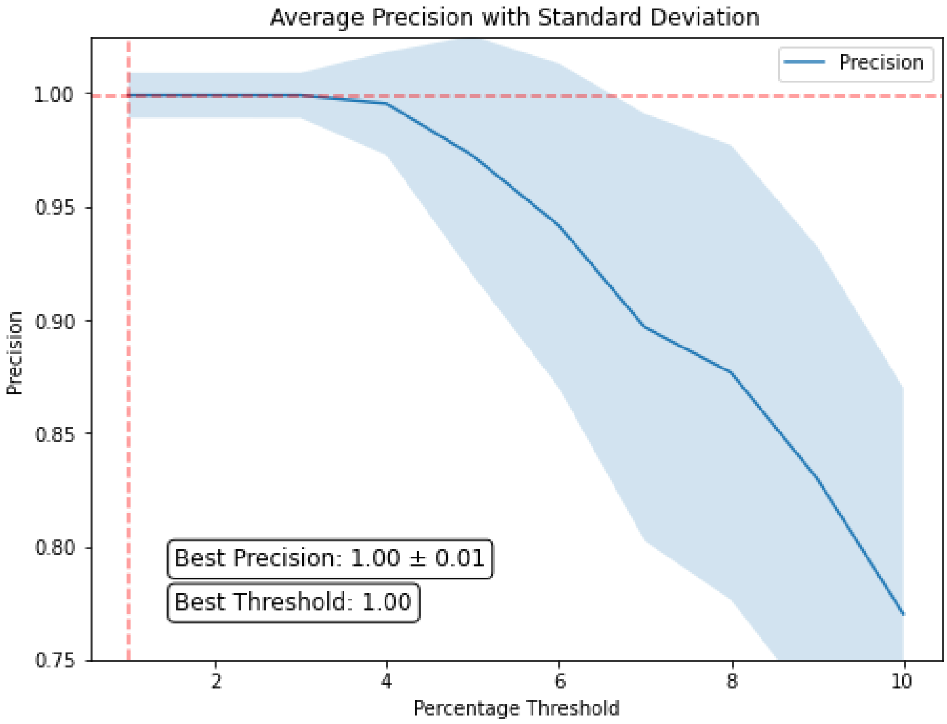

Figure 4.4: Average Precision of 1,000 Composite Models Across Different Percentage Thresholds Using Cited Default Values

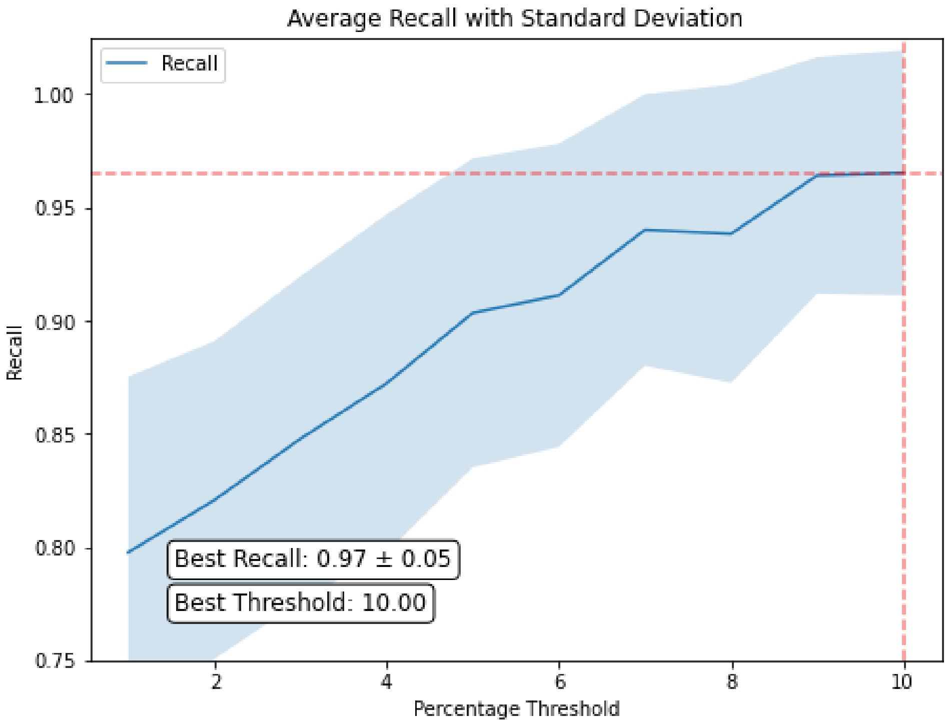

Figure 4.5: Average Recall of 1,000 Composite Models Across Different Percentage Thresholds Using Cited Default Values

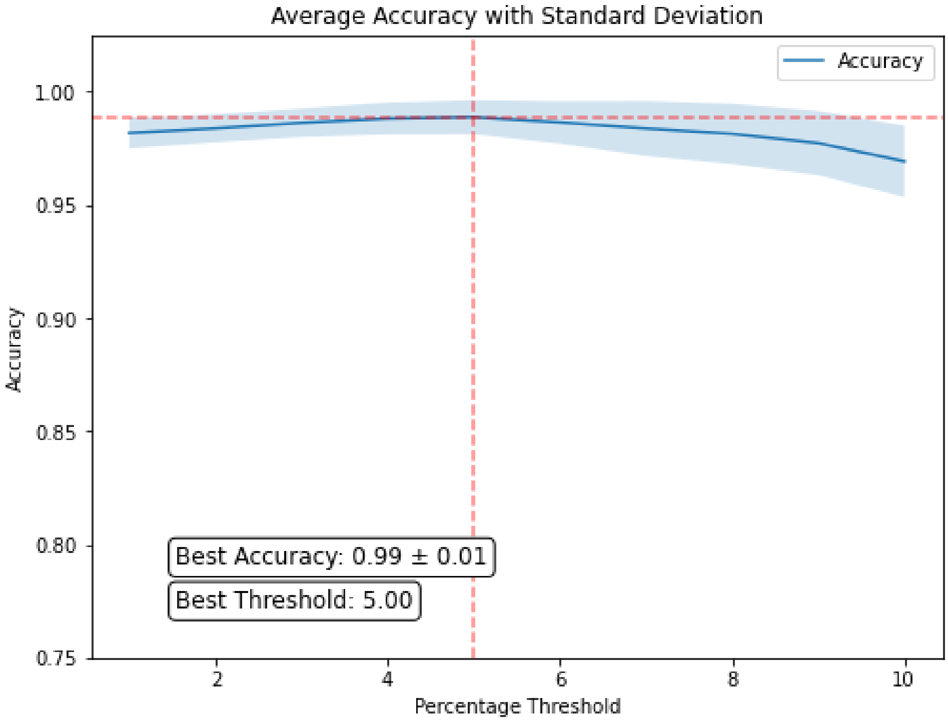

Figure 4.6: Average Accuracy of 1,000 Composite Models Across Different Percentage Thresholds Using Cited Default Values

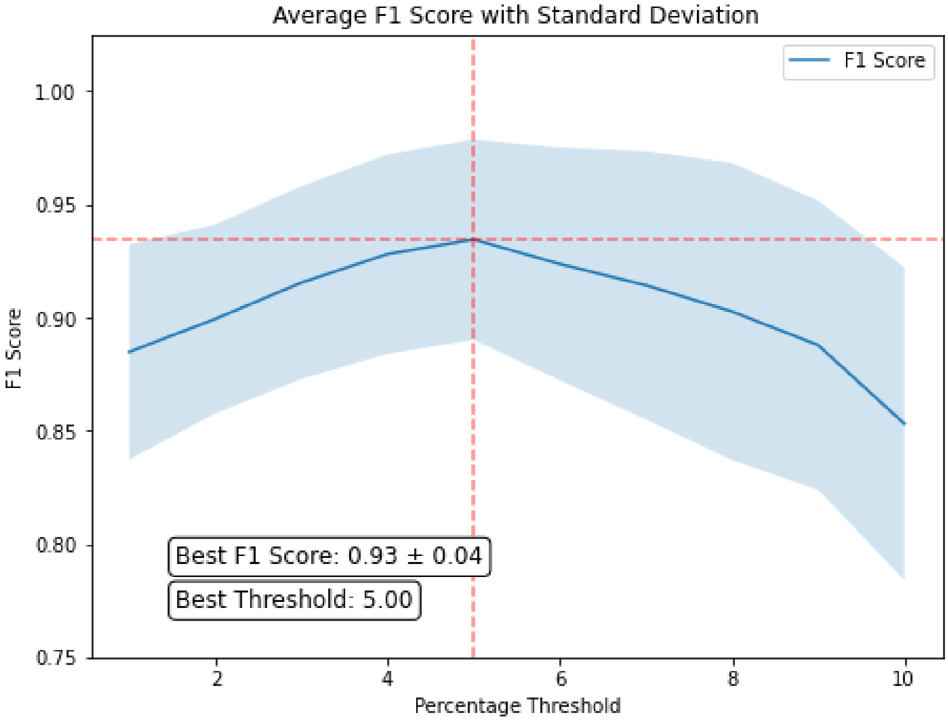

Figure 4.7: Average F1-Score of 1,000 Composite Models Across Different Percentage Thresholds Using Cited Default Values

5 Comprehensive Conclusive Discussion

5.1 Discussion Results: A Fusion of Doc2Vec and Isolation Forest

In the course of this specific implementation, an approach was explored that fused the Doc2Vec vectorization model component with the Isolation Forest algorithm model component. This fusion aimed to combine the strengths of both techniques, creating a robust and efficient model.

A key part of this specific setup is the adjustable parameter 'percentage_threshold', which can be changed from 0 to 100. The research showed an interesting link between this parameter and how well the model performs, measured by Precision, Accuracy, F1-score, and Recall. When increasing this parameter, Precision went down, but Recall went up. On the other hand, a lower parameter value improved Precision, but made Recall worse. **Notably**, when setting 'percentage_threshold' to **5**, both the F1-score and Accuracy were at their best, with an accuracy of **99%** and an F1-score of **93%**.

Assuming a potential use case involving 1,000,000 log entries and 75,000 semantic anomalies, the outcome would be, from a relative perspective, similar to the above-shown results if Accuracy and F1-Score remain consistent. In such a hypothetical example, the average results would be as follows:

Doc2Vec & Isolation Forest	**Predicted Log Entries**		
Actual Log Entries	**Anomaly**	**Normal**	
Anomaly	True Anomaly (67.500)	False Normal (7.500)	75.000
Normal	False Anomaly (1.730)	True Normal (923.270)	925.000
	69.230	930.770	1.000.000

Table 5.1: Confusion Matrix - Hypothetical Use Case - Percentage Threshold Set To 5

Conclusively derived: In addition to solely relying on manual human analysis, this composite model, a combination of Doc2Vec and the Isolation Forest algorithm, shows potential in the field of semantic anomaly detection being able to assist in identifying and prioritizing critical issues by condensing log data into smaller subsets, primarily consisting of semantic anomalies.

5.2 Possible Alternative Composite Models

As this approach relies on two different model components - a vectorization component and an anomaly detection component - there are many options that could be combined as well, such as:

5.2.1 Vectorization Techniques

1. **GloVe (Global Vectors for Word Representation):**

   > "GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space." [34]

2. **ELMo (Embeddings from Language Models):** A deep contextualized word representation technique that captures both syntactic and semantic information. ELMo embeddings are generated from a pre-trained bidirectional language model, providing contextualized vector representations for words in a sentence. [35, p.2f]

3. **FastText:** A vectorization technique that extends the Word2Vec approach by considering subword information [23]. FastText generates vector representations for words and phrases by taking into account character n-grams, making it particularly effective for morphologically rich languages. [29]

4. It is noteworthy that various pre-trained vectorization models are available for use as well, such as **Google's BERT** [7] or **OpenAI's ChatGPT** [32].

5.2.2 Anomaly Detection Techniques (requiring vector input)

1. **Local Outlier Factor (LOF):** A density-based anomaly detection algorithm

   > "used for Unsupervised outlier detection. It produces an anomaly score that represents data points which are outliers in the data set. It does this by measuring the local density deviation of a given data point with respect to the data points near it." [11]

2. **One-Class SVM (Support Vector Machine):** An unsupervised learning algorithm that detects outliers by learning a decision boundary around the normal data points. [5]

3. **K-means Clustering:** A clustering algorithm that partitions data into K clusters based on similarity. Anomalies can be detected by identifying data points that are distant from their assigned cluster centroids. [26]

By combining different vectorization techniques with suitable anomaly detection methods, many different composite models for detecting semantic anomalies can be created.

5.3 General Limitations of such Composite Models in Log Message Environments

1. *Noisy Data*: Log messages can be noisy and unstructured, which may affect the performance of the vectorization model component and, indirectly, the anomaly detection model component.
2. *High Dimensionality*: The vectorization process may result in high-dimensional data, which can lead to increased computational complexity and longer training times for the anomaly detection model component.
3. *Scalability*: As the volume of log messages increases, this approach may face scalability issues in terms of processing time and memory requirements.
4. *Parameter Tuning*: The performance of this approach depends on the choice of parameters for both the vectorization model component and the anomaly detection model component. Finding the optimal parameters can be challenging and time-consuming.
5. *False Positives/Negatives*: Generally, this approach may produce false positives (detecting normal events as anomalies) or false negatives (failing to detect actual anomalies), depending on the chosen model components and their respective parameters.
6. *Limited Interpretability*: Given that a composite model is generated, it may produce results that are difficult to interpret. This makes it challenging to understand the underlying reasons for detected anomalies.
7. *Adaptability*: The model components may not quickly adapt to changes in log message patterns after new system configurations are implemented. This could require retraining of the model components to maintain accurate anomaly detection; otherwise, it might lead to a higher rate of false positives (detecting normal events as anomalies).
8. *Dependency on Training Data*: Furthermore, the performance of this approach depends on the quality and representativeness of the training dataset. If the training data does not adequately represent the range of normal log messages, this approach may not perform well in detecting semantic anomalies.
9. *Transferability*: A trained model may be too specific to the system it was trained on, making it difficult to apply to other systems without generating a new model. This limits the model's ability to generalize and be used effectively across different log message environments.

5.4 Final Conclusion

The effective combination of Doc2Vec and the Isolation Forest algorithm illustrates that systems can be developed consisting of two elements: one for converting text data into vectors that preserve semantic meaning (vectorization), and another for detecting atypical vectors (anomaly detection). These components can adapt individually to systems generating log messages, learning their distinct patterns. Although such patterns are influenced not only by the system's configuration but also by the diverse ways each user interacts with the system, they can be comprehended by composite models like the one presented, underscoring their wide-ranging applicability. **In addressing the initial research question, the findings presented in this project thesis confirm previous and similar studies, highlighting the overall effectiveness of such composite models, and ultimately leading to an effective approach for automatically detecting semantic anomalies within log messages.**

5.5 Outlook

In the pursuit of advancing the automatic detection of semantic anomalies in log messages, future research should primarily focus on three key aspects:

1. The first aspect emphasizes the importance of creating and maintaining accessible, validated and representative open-source datasets derived from single systems containing log entries. These datasets should be labeled, organized into consecutive weekly intervals, made available separately for multiple systems, ideally with large file sizes, and also split into test and train data. At the moment there is no such labeled data publicly available for operating systems such as Linux, Microsoft and Mac OS.
2. The second aspect involves exploring and comparing various combined vectorization and anomaly detection techniques to identify the most effective composite model.
3. The third aspect would focus on the question of what level of data specificity is required in terms of the systems included in the training. How would it be measured whether it is better to combine the log data from different systems or to train composite models for each individual system? Could it even be beneficial to use the data intended for testing also as training data, rather than attempting to find a general composite model, and instead create a composite model that is specifically applied to its own training data?

These three key points are able to contribute to ensure a reliable environment for model development, unbiased performance assessment, and a deeper understanding of the strengths and limitations of (alternative) composite models, ultimately leading to more effective approaches for automatically detecting semantic anomalies within log messages.

6 Code

https://github.com/FxbxxnSxhxxrxxxx/semantic_anomaly_detection.git

Bibliography

[1] Inc. Amazon Web Services. *What are Log Files? - Log Files Explained - AWS*. Accessed: 2023-08-23. 2023. URL: `https://aws.amazon.com/what-is/log-files/?nc1=h_ls`.

[2] Abdelhadi Azzouni. *Anomaly detection in log sequences — Log analysis with PacketAI (Part 3)*. Accessed: 2023-08-23. Jan. 2022. URL: `https://medium.com/packet-ai/anomaly-detection-in-log-sequences-log-analysis-with-packetai-part-3-ready-bb48a9e4cbd8`.

[3] upGrad blog. *Top 10 Reasons Why Python is So Popular With Developers in 2023*. Accessed: 2023-09-05. 2023. URL: `https://www.upgrad.com/blog/reasons-why-python-popular-with-developers/`.

[4] Citizix. *How to Setup Central Logging Server with Rsyslog in Ubuntu 20.04*. Accessed: Sep. 6, 2023. 2022. URL: `https://citizix.com/how-to-setup-central-logging-server-with-rsyslog-in-ubuntu-20-04/`.

[5] Baeldung on Computer Science. *What Is One Class SVM and How Does It Work?* Accessed: 2023-09-12. 2022. URL: `https://www.baeldung.com/cs/one-class-svm`.

[6] *Data Science in Cybersecurity — Accuracy vs. Efficiency*. Accessed: 2023-09-26. 2023. URL: `https://dataforest.ai/blog/data-science-in-cybersecurity-the-guardian-at-the-gates`.

[7] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: (2018). URL: `https://arxiv.org/pdf/1810.04805`.

[8] Markus Fält, Stefan Forsström, and Tingting Zhang. "Machine Learning Based Anomaly Detection of Log Files Using Ensemble Learning and Self-Attention". In: *2021 5th International Conference on System Reliability and Safety (ICSRS)*. 2021, pp. 209–215. DOI: `10.1109/ICSRS53853.2021.9660694`.

[9] Amir Farzad and T. Aaron Gulliver. "Unsupervised log message anomaly detection". In: *ICT Express* 6.3 (2020), pp. 229–237. ISSN: 2405-9595. DOI: `https://doi.org/10.1016/j.icte.2020.06.003`. URL: `https://www.sciencedirect.com/science/article/pii/S2405959520300643`.

[10] Ralph Foorthuis. "On the nature and types of anomalies: a review of deviations in data". In: *International journal of data science and analytics* 12.4 (2021), pp. 297–331. DOI: `10.1007/s41060-021-00265-1`.

[11] GeeksforGeeks. *Local Outlier Factor*. Accessed: 2023-09-12. 2020. URL: `https://www.geeksforgeeks.org/local-outlier-factor/`.

[12] GeeksforGeeks. *Python RegEx*. Accessed on 2023-09-05. 2020. URL: `https://www.geeksforgeeks.org/python-regex/`.

[13] Gensim. *Topic Modelling for Humans*. Accessed on 2023-09-05, Updated on 2022-12-21. 2022. URL: `https://radimrehurek.com/gensim/intro.html#what-is-gensim`.

[14] Gensim. *Topic modelling for humans*. Accessed: 2023-08-28. 2022. URL: `https://radimrehurek.com/gensim/models/doc2vec.html`.

[15] Aarish Grover. *Anomaly Detection for Application Log Data*. Accessed: 2023-08-31. 2023. URL: https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1640&context=etd_projects.

[16] Shilin He et al. "Experience Report: System Log Analysis for Anomaly Detection". In: *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. 2016, pp. 207–218. DOI: 10.1109/ISSRE.2016.21.

[17] Abdulkader Helwan and Dilber Uzun Ozsahin. "Sliding Window Based Machine Learning System for the Left Ventricle Localization in MR Cardiac Images". In: *Applied Computational Intelligence and Soft Computing* 2017 (2017), pp. 1–9. DOI: 10.1155/2017/3048181.

[18] Fraunhofer Institute for Industrial Mathematics ITWM. *High Performance Computing*. Accessed: 2023-08-25. 2023. URL: https://www.itwm.fraunhofer.de/en/departments/hpc.html.

[19] Boehringer Ingelheim. *Boehringer Ingelheim - Value Through Innovation*. Accessed: 2023-08-28. 2021. URL: https://www.boehringer-ingelheim.com/.

[20] Boehringer Ingelheim. *Boehringer Ingelheim Achieves Up to Five Times Faster Delivery to Market with Red Hat OpenShift*. Accessed: 2023-09-12. 2021. URL: https://www.redhat.com/en/about/press-releases/boehringer-ingelheim-achieves-five-times-faster-delivery-market-red-hat-openshift.

[21] Boehringer Ingelheim. *Geschäftsbericht 2021*. Accessed: 2023-09-12. 2022. URL: https://annualreport.boehringer-ingelheim.com/2021/magazine/quantum-computing.html.

[22] Christian Janiesch, Patrick Zschech, and Kai Heinrich. "Machine learning and deep learning". In: *Electron Markets* 31.3 (2021), pp. 685–695. DOI: 10.1007/s12525-021-00475-2.

[23] Huang Kung-hsiang. *Word2Vec and FastText Word Embedding with Gensim - Towards Data Science*. Accessed: 2023-09-11. 2018. URL: https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c.

[24] Jey Han Lau and Timothy Baldwin. *An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation*. Accessed: 2023-08-28. 2016. URL: https://arxiv.org/pdf/1607.05368.pdf.

[25] Quoc V. Le and Tomas Mikolov. *Distributed Representations of Sentences and Documents*. Accessed: 2023-08-28. 2014. URL: https://arxiv.org/pdf/1405.4053.pdf.

[26] Dino Little. *Outlier Detection Using K-means Clustering In Python*. Accessed: 2023-09-12. Towards Dev. 2022. URL: https://towardsdev.com/outlier-detection-using-k-means-clustering-in-python-214188fc90e8.

[27] Samuele Mazzanti. *"Isolation Forest": The Anomaly Detection Algorithm Any Data Scientist Should Know*. Accessed: 2023-08-28. July 2021. URL: https://towardsdatascience.com/isolation-forest-the-anomaly-detection-algorithm-any-data-scientist-should-know-1a99622eec2d.

[28] McKinsey. *Pursuing quantum computing in pharma with purpose*. Accessed: 2023-08-25. 2023. URL: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/pursuing-quantum-in-pharma-with-purpose-an-interview-with-boehringer-ingelheims-cto.

[29] Tomas Mikolov. *FAIR open-sources fastText*. Accessed: 2023-09-11. 2016. URL: https://engineering.fb.com/2016/08/18/ml-applications/fair-open-sources-fasttext/.

[30] *Most Influential Papers*. Accessed: 2023-09-13. ISSRE 2021. 2022. URL: http://wut-dscl.cn/issre/MostInfluentialPapers.html.

[31] NumPy. *NumPy*. Accessed on 2023-09-05, Updated on 2023-09-02. 2023. URL: https://numpy.org/about/.

[32] OpenAI. *OpenAI Platform*. Accessed: 2023-09-12. 2023. URL: https://platform.openai.com/docs/guides/embeddings.

[33] pandas. *Python Data Analysis Library*. Accessed on 2023-09-05, Updated on 2023-09-05. 2023. URL: https://pandas.pydata.org/about/.

[34] Jeffrey Pennington. *GloVe: Global Vectors for Word Representation*. Accessed: 2023-09-11. 2021. URL: https://nlp.stanford.edu/projects/glove/.

[35] Matthew E. Peters et al. *Deep contextualized word representations*. Accessed: 2023-09-11. 2018. URL: https://arxiv.org/pdf/1802.05365.

[36] Piotr Ryciak, Katarzyna Wasielewska, and Artur Janicki. "Anomaly Detection in Log Files Using Selected Natural Language Processing Methods". In: *Applied Sciences* 12.10 (2022), p. 5089. DOI: 10.3390/app12105089.

[37] Bertil Schmidt and Andreas Hildebrandt. "Next-generation sequencing: big data meets high performance computing". In: *Drug Discovery Today* 22.4 (2017), pp. 712–717. DOI: 10.1016/j.drudis.2017.01.014.

[38] scikit-learn. *Machine Learning in Python — scikit-learn 1.3.0 documentation*. Accessed on 2023-09-05, Updated on 2023-08-29. 2023. URL: https://scikit-learn.org/stable/getting_started.html.

[39] scikit-learn. *sklearn.ensemble.IsolationForest*. Available online at https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html, Accessed: 2023-08-28. 2023.

[40] Yangyi Shao et al. "Log Anomaly Detection method based on BERT model optimization". In: *2022 7th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*. 2022, pp. 161–166. DOI: 10.1109/ICCCBDA55098.2022.9778900.

[41] Li Sun et al. *Detecting Anomalous User Behavior Using an Extended Isolation Forest Algorithm: An Enterprise Case Study*. Accessed: 2023-08-31. 2016. URL: https://arxiv.org/pdf/1609.06676.pdf.

[42] İRem Tanrıverdi. *Model Evaluation Metrics in Machine Learning - Analytics Vidhya - Medium*. Accessed: 2023-08-25. Apr. 2021. URL: https://medium.com/analytics-vidhya/model-evaluation-metrics-in-machine-learning-928999fb79b2.

[43] Zhi-Hua Zhou. *Isolation Forest*. Accessed: 2023-08-28. 2008. URL: https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf.

[44] Jieming Zhu et al. *Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics*. Accessed: 2023-09-13. 2020. URL: https://arxiv.org/pdf/2008.06448.pdf.

[45] Zhuping Zou et al. "A Docker Container Anomaly Monitoring System Based on Optimized Isolation Forest". In: *IEEE Trans. Cloud Comput.* 10.1 (2022), pp. 134–145. DOI: 10.1109/TCC.2019.2935724.