Masterarbeits

Structure embeddings for OpenSSH heap dump analysis

A report by

Lahoche, Clément Claude Martial

Prüfer

Prof. Dr. Michael Granitzer

Christofer Fellicious

Prof. Dr. Pierre-Edouard Portier

November 3, 2023

Abstract

In the dynamic landscape of digital technology, the role of cybersecurity as a cornerstone of IT systems has become increasingly prominent. The escalating complexity of cyber threats has brought digital forensics to the forefront, particularly in the examination of heap dumps from main memory. The Secure Shell protocol (SSH), essential for secure communication, serves a dual purpose: it is a line of defense and a potential channel for malicious activities. This research hones in on predicting SSH keys in OpenSSH memory dumps to strengthen security against unauthorized entry and to aid in the creation of advanced security solutions like honeypots.

This Master's thesis is a key component of the SmartVMI project, which is focused on improving AI-powered methodologies for detecting attacks and conducting digital forensics. The thesis investigates the development of data embeddings specifically for analyzing OpenSSH heap dumps and identifying SSH keys, striving for efficacy and uniformity across different OpenSSH versions and uses through machine learning and deep learning techniques. It showcases a broad spectrum of embedding methods, including those based on graphs, statistics, and natural language processing, and evaluates their effectiveness and consistency.

The research builds upon the foundational work of SSHkex [30] and SmartKex [9], enhancing their methods and outcomes and exploring the possibilities of emerging approaches not yet fully explored.

Spanning from October 2022 to October 2023, this document narrates the progression of a year-long Master's thesis research conducted within the PhDTrack program, a collaboration between the University of Passau and INSA Lyon. Supervised by Prof. Dr. Michael Granitzer and Christofer Fellicious from the University of Passau, and Prof. Dr. Pierre-Edouard Portier from INSA Lyon, the thesis offers a thorough examination of the latest advancements in key prediction for OpenSSH memory dumps. It articulates the research inquiries, experimental setups, development of programs, and the results, while also considering potential directions for future research.

Acknowledgements

My heartfelt thanks are extended to Christofer Fellicious, my esteemed supervisor at the University of Passau, for his invaluable mentorship and support throughout my research journey.

I am profoundly grateful to my colleague and dear friend, Florian Rascoussier, for his unwavering technical support and moral encouragement. His partnership in various aspects of this thesis has been instrumental to my Master's experience.

I would also like to express my sincere appreciation to my supervisor at INSA Lyon, Prof. Pierre-Edouard Portier, and to Prof. Dr. Michael Granitzer at the University of Passau. Their insights and

guidance have significantly enriched my research and academic development.

A special acknowledgment is due to the individuals who have supported me in this endeavor, notably Lionel Brunie, Director of the Computer Science Department at INSA Lyon, who has enabled the PhDTrack program from the French side, and Harald Kosch, Head of the Chair of Distributed Information Systems at the University of Passau, who has facilitated this program from the German perspective.

I am thankful to Elöd Egyed-Zsigmond, PhDTrack coordinator at INSA Lyon, for his assistance with subject selection and administrative matters, and to Natalia Lucari, also a PhDTrack coordinator at INSA Lyon, for her steadfast support and assistance throughout the program. My gratitude also goes to Ophelie Coueffe, the PhDTrack coordinator at the University of Passau, for her aid and support during this academic journey.

I am appreciative of all my fellow PhDTrack students for fostering a collaborative and intellectually stimulating environment, and for the rich discussions that have marked our shared journey over the past two years.

Finally, I must express my heartfelt thanks to my family, who have been my constant source of love and encouragement throughout the entirety of this Masterarbeit.

Contents

List of Figures

List of Tables

Algorithms and program code

1 Introduction

Digital forensics is a linchpin in cybersecurity, enabling the extraction of vital evidence from devices like PCs. This evidence is key for detecting malware and tracing intruder activities. Analyzing a device's main memory is a go-to technique in this field. The fusion of machine learning promises to amplify and streamline these analyses.

With the rising need for encrypted communication, Secure Shell (SSH) protocols are now commonplace. However, these security-focused channels can inadvertently shield malicious actions, posing challenges to standard investigative approaches. Cutting-edge research offers solutions. The work in *SmartKex: Machine Learning Assisted SSH Keys Extraction From The Heap Dump* [9] highlights how machine learning can boost the extraction of session keys from OpenSSH memory images. In a complementary vein, „SSHkex: Leveraging virtual machine introspection for extracting SSH keys and decrypting SSH network traffic“ [30] showcases the power of Virtual Machine Introspection (VMI) for direct SSH key extraction.

Inspired by *SmartKex: Machine Learning Assisted SSH Keys Extraction From The Heap Dump*, this thesis zeroes in on a central challenge: data embedding. While previous studies set the stage for key extraction, the data embedding technique, especially windowing, can be optimized. The design of data embeddings is pivotal for machine learning efficacy, especially in nuanced tasks like memory analysis. This research introduces fresh embedding strategies, aiming to refine extraction and unearth deeper memory snapshot patterns. Merging graph embeddings with advanced machine learning, the goal is to craft a sophisticated toolkit for OpenSSH heap dump studies, bridging digital forensics and machine learning.

2 Research Questions

- What are the most effective techniques for embedding byte sequences, especially when aiming to extract structures containing SSH keys for machine learning purposes?

- Do the embeddings designed show noticeable differences based on the various applications of OpenSSH, considering the wide range of SSH key sizes and the complex operations of OpenSSH?

- How can we ensure the consistency and stability of these embeddings across the wide variety of OpenSSH versions and usages?

3 Structure of the Thesis

The structure of this thesis is organized into several distinct parts, each serving a specific purpose in the presentation of the research. The **Introduction** sets the stage, providing an overview and the objectives of the thesis. Following this, the **Related Work** section delves into the origins of the thesis, offering context and acknowledging the contributions that have influenced this research. The **Background** section lays the technical foundation, explaining the concepts and technologies that underpin the thesis.

The **Methods** section is comprehensive, encompassing several key areas of exploration: the **Dataset Exploration** sub-section details the data under scrutiny; the **Embedding Techniques Used** sub-section describes the methods employed to process and analyze the data; the **Embedding Test on Performance** and **Embedding Test on Coherence** sub-sections evaluate the effectiveness of these techniques.

The **Results** section presents the findings of the research, while the **Discussion** section reflects on the implications, limitations, and potential areas for further inquiry. Finally, the **Conclusion** summarizes the thesis, reaffirming its contributions and outcomes. Supporting materials and additional information are provided in the **Appendix**, which serves as a reference and complements the main text.

4 Related work

The embedding of memory heap dumps for the detection of SSH keys is a niche yet crucial area of research, especially in the context of cybersecurity and digital forensics. This section reviews two seminal papers that have significantly influenced our work: *SSHKex*, which delves into Virtual Machine Introspection (VMI) and memory forensics, and *SmartKex*, which employs machine learning techniques for SSH key detection.

4.1 Virtual Machine Introspection and Memory Forensics: SSHKex

SSHKex is an initiative that delves into the intricacies of analyzing encrypted SSH traffic. By harnessing the capabilities of VMI, Sentanoe and Reiser spearheaded this project to discreetly extract SSH keys and decrypt SSH network communications, ensuring the preservation of evidence [30].

The methodology adopted by **SSHKex** seamlessly integrates conventional network traffic monitoring with dynamic SSH session key retrieval. A pivotal assumption is the familiarity with the SSH server's implementation, which is vital for the extraction process. Tools such as LibVMI and Volatility, under the VMI umbrella, are employed to provide an unaltered perspective of the guest VM's state, enabling the efficient pinpointing of SSH session keys within a Linux system's primary memory.

Outlined below is the **SSHKex** key extraction procedure:

1. **Data Structure Insights**: The technique capitalizes on in-depth understanding of the data structures housing the keys. Debugging symbols, tailored to the SSH version on the target, offer crucial offset values, aiding in key material extraction. Key structures encompass `struct ssh`, `struct session_state`, `struct newkeys`, and `struct sshenc`, which collectively house details like IP addresses, session statuses, and encryption keys.

2. **OpenSSH Function Tracing**: This step involves tracing functions to accurately locate data structures and timely key extraction. Emphasis is on `kex_derive_keys` (for key generation initiation) and `do_authentication2` (triggering user authentication).

3. **Breakpoint Implementation**: For debugging purposes, software breakpoints are strategically embedded in the program's execution. Using VMI, SSHKex introduces these breakpoints at the starting points of the two pivotal functions mentioned above.

4. **Extraction of Keys**: The `kex_derive_keys` function's invocation prompts SSHKex to initially capture the `ssh struct`'s address. The subsequent call to the `do_authentication2` function facilitates the extraction of actual keys, adhering to recognized structures.

5. **Key Classification**: OpenSSH designates distinct indices in the `newkeys` structure for client-to-server and vice versa keys. SSHKex's extraction is based on these specific indices.

6. **Managing Multiple Sessions**: OpenSSH handles numerous SSH sessions by initiating child processes. SSHKex broadens its extraction approach to each child process, identifying them via their distinct process IDs.

A standout feature of **SSHKex** is its commitment to discretion, conservation, and maintaining evidence authenticity. The methodology is crafted to minimize intrusiveness, ensuring no alterations to the scrutinized system. This is paramount in forensic scenarios where evidence sanctity is of utmost importance.

4.2 Machine Learning for SSH Key Detection: SmartKex

SmartKex builds upon the foundation of extracting SSH keys from heap memory dumps, aiming to streamline and automate the process. The project stands out by integrating machine learning techniques, enhancing the efficiency and precision of key extraction. This contrasts with the more complex SSHKex approach, which necessitates in-depth SSH knowledge and breakpoint injections.

SmartKex proposes two key extraction methods:

- *Brute-Force Baseline Method*: A conventional method that sifts through heap memory, identifying potential keys based on recognized patterns.
- *Machine Learning-Assisted Method*: Utilizes a Random Forest algorithm, trained on an imbalanced dataset balanced using SMOTE. While this method offers high precision and recall, it's probabilistic, making it less exact than the brute-force approach.

4.2.1 Baseline Brute-Force Method

The brute-force approach of **SmartKex** encompasses the following steps [9]:

1. *Heap Dump Creation*: Binary files of the OpenSSH server process are generated (methodology unspecified in SmartKex) and are presumed to be based on a linux-x86_64 architecture.
2. *Data Trimming*: The method trims irrelevant memory pages based on Hamming distance to reduce heap size.
3. *Key Search*: The algorithm scans the heap, considering a 128-byte length as a potential key, iterating until the heap's end.
4. *Decryption Trials*: Each potential key undergoes decryption attempts on network packets. Failed attempts lead to the next potential key.

Despite its exactness, the brute-force method is resource-intensive and is less efficient when keys are towards the heap dump's end.

4.2.2 Machine Learning-Assisted Method

SmartKex's innovation lies in its machine learning methodology, which, while sacrificing exactness, offers speed and accuracy. The method also reduces the heap to under 2% of its original size. The steps include:

1. *Heap Dump Inputs*: As with the brute-force method, binary files from OpenSSH are the primary inputs.
2. *Data Preprocessing*: The heap dump is reshaped into an N $\times$ 8 matrix. High entropy sections, potential encryption keys, are flagged using logical operations on byte differences.
3. *Model Training*: A Random Forest algorithm is trained on 128-byte segments of the processed heap. Given the dataset's imbalance, a stacked classifier approach is employed.
4. *Key Detection*: Predictions on potential key-containing slices are made using the model, followed by a brute-force extraction.

SmartKex not only outperforms the brute-force method in speed but also excels in accuracy. Its applications span across cybersecurity and memory forensics. The adaptability of its machine learning methodology makes it a valuable asset for both researchers and professionals. The project's open-source nature further enhances its accessibility, with the code available on GitHub.

4.3 Our Contribution

In this research, we delve deep into the realm of SSH key detection by exploring multiple embedding techniques: graph-based, statistical-based, and deep learning-based embeddings. Our approach is twofold. Firstly, we employ these embeddings in conjunction with a classifier model, comparing their performance to determine the most effective method for SSH key extraction from memory heap dumps. Secondly, to address the challenges of consistency across various OpenSSH versions and usages, we implement a clustering approach. This ensures that our embeddings not only accurately detect SSH keys but also maintain coherence and adaptability across different OpenSSH environments.

5 Background

Building upon our exploration of SSH key detection, we recognize that OpenSSH plays a pivotal role in secure communication. The heap dumps of OpenSSH, essentially memory snapshots, are rich reservoirs of data. By generating graphs from these dumps, intricate connections between data structures, identified via their malloc headers, and their corresponding pointers are revealed.

Our research delves into the intelligent embedding of connections derived from OpenSSH heap dumps. Beyond mere graph representations, it's essential to comprehend the raw byte sequences within these dumps. Traditional methods such as Shannon entropy, Byte Frequency Distribution (BFD), and bigram frequencies form the foundation. However, the rise of deep learning has ushered in a range of sophisticated techniques. Models like Recurrent Neural Networks (RNN) [20] (including Long Short-Term Memory (LSTM)[14] and Gated Recurrent Units (GRU)[5]), sequence-to-sequence learning [32], and convolutional approaches (CNN)[21] offer novel perspectives on raw byte embedding. The integration of CNN with recurrent networks has proven effective in sequence modeling [2]. Furthermore, neural networks, especially in lossless representations, have shown promise in file fragment classification [12]. Our exploration also encompasses the potential of transformers[34] and autoencoders.

To identify structures containing OpenSSH keys, we employ machine learning classifiers. Among the standard models, the Random Forest classifier stands out for its versatility and accuracy. Balancing strategies are crucial, especially when dealing with imbalanced datasets, to ensure that the classifier doesn't exhibit bias towards the majority class. In addition to classification, our research emphasizes clusterization techniques to group similar data points and ensure coherence. Algorithms such as DBSCAN and OPTICS are at the forefront of our clusterization approach, providing insights into the inherent groupings within the OpenSSH heap dump data.

The aim of this background section is to provide a comprehensive overview of graph creation from heap dumps, techniques for raw byte embedding, and their role in identifying OpenSSH key structures. By merging age-old techniques with modern approaches, we strive to highlight the most effective methods for analyzing OpenSSH heap dump.

5.1 Traditional Statistical Embedding

Within the domain of machine learning, how data is represented significantly impacts the performance of models. Even though traditional statistical embedding techniques have been around before many contemporary methods, they continue to be vital in readying data for machine learning endeavors. Rooted in statistical foundations, these techniques provide a methodical approach to transform raw data into concise and meaningful forms. In this subsection, we'll delve into the nuances of entropy and its role in byte sequence embedding, Byte Frequency Distribution (BFD), and also highlight other classical statistical embedding methods pivotal in data representation for machine learning.

5.1.1 Entropy and its application in byte sequence embedding

Entropy, a fundamental concept in information theory, quantifies the amount of uncertainty or randomness associated with a set of data. Introduced by Claude Shannon in his groundbreaking work [31], entropy serves as a measure of the average information content one can expect to gain from observing a random variable's value.

Mathematically, the entropy $H(X)$ of a discrete random variable X with possible values $\{x_1, x_2, \ldots, x_n\}$ and probability mass function $P(X)$ is given by:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i) \tag{5.1}$$

Within the scope of identifying SSH keys, the significance of entropy cannot be understated. Byte sequences exhibiting high entropy typically reflect a multifaceted and varied informational content, traits that are synonymous with encryption keys, especially those in SSH. Sequences with pronounced entropy are often prime contenders for SSH keys due to their inherent randomness and lack of predictability, mirroring the attributes of robust security keys.

Fundamentally, entropy acts as a quantitative tool to evaluate the depth of information within data. When applied to SSH, it suggests that data sequences with elevated entropy levels have a heightened probability of correlating with secure keys. This positions entropy as an essential instrument for pinpointing and authenticating SSH keys.

5.1.2 Byte Frequency Distribution (BFD)

In the complex world of raw byte embedding, Byte Frequency Distribution (BFD) and n-gram embedding stand out as essential methods, each bringing unique benefits to data representation. BFD zeroes in on the distribution of individual byte values in a raw byte sequence. Analyzing these distributions allows for the identification of patterns that reflect the inherent nature of the data. This embedding technique becomes particularly relevant when assessing the randomness or structure of byte sequences, such as when detecting encrypted data or pinpointing specific file signatures.

On the other hand, n-gram embedding dives deeper into raw byte sequences. Instead of focusing solely on individual bytes, it captures patterns formed by sequences of 'n' consecutive bytes. This approach garners a wider range of contextual information from the raw byte data. For example, a trigram (3-gram) examines patterns formed by three sequential bytes, providing a richer representation than single byte values. Yet, a challenge with n-gram embedding is the potential for the output vector size to grow exponentially as 'n' increases, posing computational and storage issues, especially in real-time scenarios.

In the realm of raw byte embedding, both BFD and n-gram techniques offer invaluable perspectives. While BFD establishes a base representation centered on individual byte frequencies, n-gram embedding enhances it by spotlighting the complex relationships and patterns among consecutive bytes. Together, they form a robust arsenal for representing and analyzing raw byte data in a variety of applications.

5.1.3 Other traditional statistical embedding techniques

Mean Byte Value The Mean Byte Value represents the average value of all bytes in a given sequence. It provides an insight into the central tendency of the byte values in the sequence. Mathematically, for a byte sequence B of length n:

$$\text{Mean Byte Value} = \frac{1}{n}\sum_{i=1}^{n} B_i \tag{5.2}$$

Mean Absolute Deviation (MAD) MAD measures the average distance of each byte value from the mean, providing a sense of the dispersion or spread of the byte values around the mean. It is given by:

$$\text{MAD} = \frac{1}{n}\sum_{i=1}^{n} |B_i - \text{Mean Byte Value}| \tag{5.3}$$

Standard Deviation Standard Deviation quantifies the amount of variation or dispersion in the byte sequence. A higher value indicates greater variability in the byte values. It is defined as:

$$\text{Standard Deviation} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(B_i - \text{Mean Byte Value})^2} \tag{5.4}$$

Skewness Skewness[36] measures the asymmetry of the distribution of byte values around the mean. A positive value indicates a distribution that is skewed to the right, while a negative value indicates a distribution skewed to the left. It provides insights into the shape of the distribution of byte values. The Fisher's skewness[4] is :

$$\text{Skewness} = \frac{n}{(n-1)(n-2)}\sum_{i=1}^{n}\left(\frac{B_i - \text{Mean Byte Value}}{\text{Standard Deviation}}\right)^3 \tag{5.5}$$

Kurtosis Kurtosis[36] measures the "tailedness" of the distribution of byte values. A higher kurtosis value indicates a distribution with heavier tails, while a lower value indicates lighter tails. It provides insights into the extremities of the distribution. The Fisher's kurtosis[4] is :

$$\text{Kurtosis} = \frac{n(n+1)}{(n-1)(n-2)(n-3)}\sum_{i=1}^{n}\left(\frac{B_i - \text{Mean Byte Value}}{\text{Standard Deviation}}\right)^4 - \frac{3(n-1)^2}{(n-2)(n-3)} \tag{5.6}$$

n-gram on Bits When applying n-gram techniques to bits instead of bytes, we focus on sequences of 'n' consecutive bits. For example, a 2-gram on bits would consider patterns formed by two consecutive bits, resulting in four possible combinations: 00, 01, 10, and 11. This approach significantly reduces the size of the output vector compared to byte-based n-grams. By focusing on bits, we can capture more granular patterns in the data while benefiting from a more compact representation, which is computationally efficient and requires less storage.

5.2 Deep Learning Models for Raw Byte Embedding

In the area of data representation, deep learning is great for understanding raw byte sequences. Just like these models are good at understanding text, they're also good at understanding raw bytes. They can learn and show sequences on their own, which is really helpful for both text and raw bytes. In this section, we'll look at different deep learning models and how they work with raw byte embedding.

Our exploration begins with Word2Vec, a renowned technique for textual representation. Next, we'll delve into Recurrent Neural Networks (RNN). These networks, celebrated for their handling of word sequences in textual contexts, are equally adept with raw byte sequences. Moving forward, we'll discuss Convolutional Neural Networks (CNN), recognized for identifying patterns in raw bytes, a skill reminiscent of their proficiency with text. We'll subsequently introduce Autoencoders, distinguished by their distinctive learning paradigm. To round off this section, we'll examine Transformers, lauded for their expertise in capturing extended data relationships, much like their prowess with textual content.

5.2.1 Word2Vec: A Deep Dive into Word Embeddings

Word2Vec [24] is a groundbreaking algorithm in the realm of natural language processing (NLP) that transformed words into continuous vector spaces. Developed by Tomas Mikolov and his team at Google in 2013, this algorithm is based on the idea that the meaning of a word can be inferred by the context in which it appears.

5.2.1.1 Architectures

Word2Vec operates using one of two neural network architectures:

1. **Continuous Bag of Words (CBOW)**: This model predicts a target word based on its surrounding context. Given a context (a set of surrounding words), CBOW tries to predict the word that appears in the middle.

2. **Skip-Gram**: This is the inverse of CBOW. For a given word, the Skip-Gram model tries to predict the surrounding words (context). It can be visualized as predicting the 'context' from a 'word'.

5.2.1.2 Mechanism

Under the hood, Word2Vec uses a shallow neural network with a single hidden layer. However, the objective is not to achieve high accuracy in prediction but rather to learn rich word vector representations. Once trained, the weights of the hidden layer represent the word vectors.

Training proceeds as follows:

- Start with large amounts of text data and construct a vocabulary.
- For each word, create pairs of the word with its surrounding context words based on a defined window size.
- Use these pairs as input-output mappings to train the neural network.

5.2.1.3 Key Concepts and Innovations

1. **Distributional Hypothesis**: Word2Vec thrives on the idea that words appearing in similar contexts have related meanings. By analyzing vast amounts of text, the algorithm detects patterns and semantic relationships.
2. **Vector Arithmetic**: One of the most exciting outcomes of Word2Vec is its ability to perform vector arithmetic that captures semantic relationships. For example, the operation : $\text{vector}(\text{“}king''\text{)} - \text{vector}(\text{“}man''\text{)} + \text{vector}(\text{“}woman''\text{)}$ results in a vector close to $\text{vector}(\text{“}queen''\text{)}$.
3. **Subsampling and Negative Sampling**: To make training more efficient, Word2Vec introduced techniques like subsampling frequent words and negative sampling. Subsampling reduces the occurrences of high-frequency words in training, while negative sampling changes the training objective to distinguish the target word from randomly sampled 'negative' words.
4. **Word Analogies**: Word2Vec embeddings capture various dimensions of word meanings, leading to the discovery of word analogies. For example, $man : woman :: king : queen$ is a typical analogy captured by Word2Vec embeddings.

5.2.1.4 Applications

Word2Vec has paved the way for many NLP applications, including:

- **Semantic Word Similarity**: Comparing the cosine similarity of word vectors provides a measure of semantic similarity.
- **Information Retrieval**: Search engines can use word embeddings to better understand query intent and document content.

- **Sentiment Analysis**: Word embeddings capture sentiment dimensions, making them useful for sentiment classification tasks.
- **Machine Translation**: Word embeddings can act as a bridge between languages, aiding in tasks like machine translation.

5.2.1.5 Conclusion

Word2Vec was a significant leap forward in the domain of NLP, offering a mechanism to capture deep semantic word relationships in a computationally efficient manner. By viewing arbitrary long byte sequences as sentences, it becomes possible to adapt the Word2Vec algorithm for purposes beyond traditional text, broadening its applicability to our context.

5.2.2 RNNs : Understanding sequence data

Recurrent Neural Networks (RNN) are great tools for text classification. They're good at understanding the deeper meanings in text. Unlike older models that use hand-made features, RNN can learn and show sequences on their own. This makes them really useful for tasks that deal with sequences. When we think about embedding raw bytes, RNN's skill in understanding sequences is similar to how they handle word sequences in text. Here is a list of different RNN models and their advantages and disadvantages.

Recurrent Convolutional Neural Network (RCNN) for Text Classification[20]: The RCNN model, as discussed in the paper by Lai et al., is designed specifically for text classification. Unlike traditional models, RCNN do not rely on handcrafted features. Instead, they employ a recurrent structure to capture contextual information about words. This approach is believed to introduce considerably less noise compared to traditional window-based neural networks. The model's bidirectional structure ensures that both preceding and succeeding contexts of a word are considered, enhancing its understanding of the word's semantics.

- **Advantages:**
  - No need for handcrafted features.
  - Captures richer contextual information.
  - less noisy.
- **Disadvantages:**
  - Complexity due to bidirectional structure.
  - Might require more computational resources.

;

Long Short-Term Memory (LSTM)[14]: The LSTM, introduced by Hochreiter and Schmidhuber, is a specialized form of RNN designed to combat the vanishing gradient problem inherent in traditional RNN. The vanishing gradient problem arises when gradients of the loss function, which are used to update the network's weights, become too small for effective learning. This typically happens in deep networks or when processing long sequences, causing the earlier layers or time steps to receive minimal updates. As a result, traditional RNN struggle to learn long-term dependencies in the data.

LSTM address this issue with their unique cell state and gating mechanisms. The cell state acts as a "conveyor belt" that can carry information across long sequences with minimal changes, ensuring that long-term dependencies are captured. The gating mechanisms, namely the input, forget, and output gates, regulate the flow of information into, out of, and within the cell. This design allows LSTMs to selectively remember or forget information, making them adept at learning and retaining long-term dependencies in sequences.

- **Advantages:**
  - Efficiently learns long-term dependencies; overcomes the vanishing gradient problem inherent in traditional RNN.
  - Often achieves faster and more stable learning.
- **Disadvantages:**
  - More complex architecture compared to basic RNN and even GRU.
  - Can be computationally intensive due to the multiple gating mechanisms.

Gated Recurrent Units (GRU)[5]: GRU are a variant of RNN that aim to capture long-term dependencies without the complexity of LSTM. They use a gating mechanism to control the flow of information, making them efficient in sequence modeling tasks.

- **Advantages:**
  - Simplified structure compared to LSTM.
  - Efficient in capturing long-term dependencies.
  - Sometimes outperforms LSTM.
- **Disadvantages:**
  - Still more complex than traditional RNN.
  - Might not always outperform LSTM in all tasks.

To sum it up, RNN are good at understanding sequences and context. This makes them a good choice for embedding raw bytes. Just like they understand words based on the words around them, RNN can find patterns in raw byte sequences, giving us a better understanding of the data.

5.2.3 CNNs : Pattern detection in raw bytes

Convolutional Neural Networks (CNN)[21] are a specialized category of deep learning models adept at identifying patterns. Originally designed for visual data, their prowess extends to tasks like image and document recognition. Drawing inspiration from the human visual cortex's biological processes, CNN are architected to autonomously and adaptively discern spatial feature hierarchies from inputs. This becomes particularly relevant when considering raw byte embedding, where the goal is to detect patterns in sequences of bytes. The CNN architecture boasts convolutional layers that perform operations on input data to capture localized patterns, and pooling layers that condense spatial dimensions while preserving crucial information. This layered approach enables CNN to detect intricate patterns by progressively building on simpler foundational patterns. When applied to byte sequences or document recognition, CNN excel, showcasing remarkable efficacy, especially in tasks like identifying patterns within raw byte sequences or recognizing handwritten content.

When tailored to CNN, the Sequence-to-Sequence (Seq2Seq)[10] approach emerges as a potent tool for transforming raw byte sequences into meaningful embeddings. The encoder segment of the Seq2Seq model is central to this transformation. It delves into the byte sequence, discerning intricate patterns and nuances, and distills this rich information into a concise context vector or embedding. This condensed representation captures the core essence of the byte sequence, positioning it as a valuable input for subsequent tasks, such as classification models.

At the heart of the encoder lie the convolutional layers, skilled in pinpointing specific patterns within the byte sequence. Whether it's unique byte combinations or indicative n-grams, these layers are primed to detect them. As they traverse the raw byte sequence, they employ specialized filters, honed to recognize these specific patterns. As the data flows through the encoder's layers, these identified patterns are synthesized and refined, culminating in a comprehensive embedding of the sequence.

Here are two Sequence-to-Sequence (Seq2Seq) models using CNN :

- **Autoencoders:** These neural network architectures[13] are designed for data compression and reconstruction. The encoder part compresses the input data into a compact representation, while the decoder reconstructs the original data from this representation. In the context of raw byte sequences, the encoder can be used to generate embeddings that capture the essential patterns and structures of the data.

- **Transformers :** Transformers[34] utilize self-attention mechanisms to weigh the significance of different parts of the input data. This allows them to capture long-range dependencies and relationships in the data. When applied to raw byte sequences, transformers can generate embeddings that consider both local and global patterns, making them particularly effective for tasks that require understanding the broader context of a sequence.

Yet, a significant challenge with traditional Sequence-to-Sequence (Seq2Seq) models using CNN is their constraint in managing inputs of varying sizes. Constructed with a set input size, they face difficulties when presented with sequences of diverse lengths, like raw byte sequences.

To address this limitation, various techniques have been employed to normalize the size of the input data. One of the most common methods is **padding**, where shorter sequences are filled with predefined placeholder values (often zeros) until they match the length of the longest sequence in the dataset. This ensures that all sequences fed into the model have a uniform length. Another approach is **bucketing**, where sequences of similar lengths are grouped together, minimizing the amount of padding required. Additionally, **truncation** can be used to shorten sequences that exceed a certain length, although this might result in the loss of some information. While these techniques enable CNN-based Sequence-to-Sequence (Seq2Seq) models to handle variable-sized inputs, it's crucial to ensure that the preprocessing steps do not introduce noise or distort the inherent patterns and relationships within the raw byte sequences.

5.2.4 The Transformer Architecture

The Transformer [34] brought a novel approach to handling sequential data without relying on traditional recurrent mechanisms. Instead, it leverages attention mechanisms to draw global dependencies between input and output, achieving state-of-the-art results in a variety of NLP tasks.

5.2.4.1 Architecture Overview

The Transformer architecture is primarily composed of an encoder-decoder structure. Both the encoder and decoder consist of a series of identical layers, each with two primary components: multi-head self-attention and a position-wise feed-forward network.

5.2.4.2 Encoder

1. **Multi-Head Self-Attention**: This mechanism allows the model to focus on different parts of the input text simultaneously. It computes attention weights for different positions in the sequence, enabling the model to capture long-range dependencies.
2. **Position-wise Feed-Forward Networks**: After the attention scores are computed, they are passed through a feed-forward network, applied independently to each position.

5.2.4.3 Decoder

The decoder has an additional layer compared to the encoder:

1. **Multi-Head Self-Attention**: Functions similarly to the encoder.

2. **Multi-Head Cross-Attention**: This attends to the encoder's output.
3. **Position-wise Feed-Forward Networks**.

5.2.4.4 Attention Mechanism

At the core of the Transformer model lies the attention mechanism, a novel approach to capturing contextual information in sequences. This mechanism facilitates the model's ability to focus variably on different parts of the input, depending on the context. This context-sensitive focusing is achieved by computing a weighted sum of values, with the weights determined by the interaction between a query and a set of key-value pairs.

The Transformer employs a variant known as the "scaled dot-product attention". This attention mechanism can be understood through the following steps:

1. **Dot Product**: For each query, its dot product with all keys is computed. This results in a score that signifies the relevance of the query with respect to each key. Higher dot product values imply higher relevance.
2. **Scaling**: The scores obtained from the dot product are then scaled down by dividing them with the square root of the dimensionality of the query and key vectors. This scaling is crucial because for larger magnitudes of the dot products, the softmax 5.3.1 function would squash its inputs causing a very small gradient – essentially a very hard softmax. Scaling counteracts this potential issue, ensuring a softer softmax where the gradient is larger.
3. **Softmax Normalization**: The scaled scores for each query are then passed through a softmax operation. This operation ensures that the scores are normalized and lie between 0 and 1, making them interpretable as probabilities. These probabilities dictate how much attention should be given to each corresponding value in the sequence.
4. **Weighted Sum of Values**: Lastly, the softmax normalized scores are used to take a weighted sum of the values. The result of this weighted sum gives the output of the attention mechanism for the given query. If a specific key is highly relevant to the query (i.e., has a high softmax score), then the corresponding value will have a larger weight in the final sum.

The beauty of this mechanism is that it allows the Transformer to consider multiple parts of the input sequence when producing an output, rather than being limited to a fixed window or relying on recurrent state. By attending variably to different input parts based on context, the Transformer can capture complex patterns and long-range dependencies in the data.

5.2.4.5 Multi-Head Attention

Instead of performing a single set of attention computations, the Transformer utilizes multiple sets (or "heads"). Each head learns different attention weights, and their outputs are concatenated and

linearly transformed to produce the final result.

5.2.4.6 Positional Encoding

In traditional recurrent architectures such as RNNs and LSTMs, the inherent recurrence mechanism enables the model to naturally take into account the order or sequence of tokens. The Transformer architecture, in stark contrast, relies entirely on self-attention mechanisms which, in their basic form, are order-agnostic. This means that without any additional information, shuffling the input tokens would produce the same output. To address this limitation and introduce the notion of order or sequence into the architecture, positional encodings are ingeniously integrated into the Transformer's design.

The positional encodings are added to the initial embeddings of tokens before they are processed by the encoder and decoder stacks. These encodings are designed such that they can provide a unique representation for each token based on its position in the sequence, regardless of the token's actual value. This ensures that the model can distinguish between tokens not just based on their content, but also based on their positions.

The positional encodings utilize sinusoidal functions, specifically sine and cosine functions, of varying frequencies. For a position p and a dimension i, the encoding is defined as:

$$PE_{(p,2i)} = \sin\left(\frac{p}{10000^{\frac{2i}{d}}}\right)$$

and

$$PE_{(p,2i+1)} = \cos\left(\frac{p}{10000^{\frac{2i}{d}}}\right)$$

where d is the dimension of the embeddings. This formulation ensures that each position produces a distinct and consistent encoding. Moreover, because of the geometric progression in the denominators, these functions create encodings that can be easily differentiated by the model across positions, providing a relative sense of position between tokens.

Using sinusoidal functions as positional encodings comes with the advantage that these encodings can be scaled to accommodate sequences of varying lengths, even beyond those seen during training. This is because the sum of the positional encoding and token embedding will result in a unique and consistent value, allowing the model to generalize to different sequence lengths. Intuitively, the sinusoidal nature of the encodings creates a pattern that the model can learn, helping it understand relative positions and distances between tokens in a sequence.

5.2.5 Benefits and Applications

- **Parallelization**: Without recurrent layers, the Transformer can process all tokens in the input sequence simultaneously, making it amenable to parallel processing and reducing training times.

- **Long-range Dependencies**: The self-attention mechanism can, in theory, relate distant words in a sequence, capturing long-term dependencies without relying on the sequence's length.

- **Flexibility**: The architecture is not inherently sequential, making it adaptable to a variety of tasks beyond NLP.

5.2.6 Conclusion

The Transformer architecture, with its focus on attention mechanisms, has set new standards in the realm of deep learning for sequential data. It serves as the foundation for various state-of-the-art models in NLP, such as BERT, GPT, and T5, underscoring its significance and versatility in the field.

5.3 Machine learning

Machine learning, an integral part of artificial intelligence, revolves around designing algorithms and statistical models that allow computers to perform tasks without being directly programmed. Instead of relying on detailed instructions for every task, machine learning techniques empower systems to learn from data and make data-driven decisions. A key method in this field is supervised learning, in which models are trained using data that comes with predefined labels. Here, each piece of data in the training set has an associated known output. The primary goal of supervised learning is to establish a relationship between inputs and outputs, enabling the model to predict or categorize new, unseen data based on this relationship.

A cornerstone in this realm is feature engineering, which involves the meticulous process of selecting and transforming variables to optimize model performance. Another challenge frequently encountered by practitioners is dealing with datasets where some classes are overrepresented, which can skew model predictions. Among the myriad of machine learning models available, certain ones have gained prominence due to their versatility and effectiveness. We will provide an overview of some of these notable models.

5.3.1 Softmax Function

The softmax function is an essential component in machine learning, especially for classification tasks. It is responsible for converting a vector of real numbers into a probability distribution.

5.3.1.1 Definition

Given a vector $\mathbf{z} = [z_1, z_2, ..., z_K]$ of real numbers, the softmax function S is defined for each element z_i as:

$$S(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{5.7}$$

where:

- e is the base of natural logarithm.
- K represents the number of classes or the dimensionality of the vector.

5.3.1.2 Intuition

1. **Exponentiation**: Each element of the vector is raised to the power of e. This ensures that all resulting elements are non-negative and exaggerates the differences between them. A significantly larger value will have an exponentially greater value compared to a smaller one after this operation.
2. **Normalization**: The exponential values are divided by their sum. This step ensures that the output values lie in the range [0, 1], and their total sums up to 1, hence forming a valid probability distribution.

5.3.1.3 Applications in Machine Learning

In classification problems, the raw outputs from a model (typically from the last layer of a neural network) are real numbers that do not necessarily sum up to one. The softmax function converts these raw scores into a probability distribution over the categories. In most cases, the category with the highest probability post the softmax operation is taken as the model's prediction.

For instance, in a neural network tasked with recognizing handwritten digits ranging from 0 to 9, the final layer might produce a 10-dimensional vector. Applying softmax on this vector gives a probability distribution over these ten digits. The digit corresponding to the highest probability after softmax is then chosen as the model's prediction.

5.3.2 Features engineering

Feature engineering[16] is a cornerstone in the realm of machine learning. It involves the artful transformation of the given feature space to optimize the performance of predictive models. The significance of feature engineering cannot be overstated; it serves as a bridge between raw data and the predictive models, ensuring that the models are fed with the most relevant and informative features. Properly engineered features can drastically reduce modeling errors, leading to more accurate and reliable predictions. Here are some of the most common feature engineering techniques:

- **Normalization and Scaling** are preprocessing techniques used to standardize the range of independent features in the data. Many machine learning algorithms, especially those that rely on distance calculations like k-means clustering or support vector machines, are sensitive to the scale of the data. If features are on different scales, one feature might dominate others, leading to suboptimal model performance. Normalization typically scales features so that they have a unit norm, while other scaling methods, such as Min-Max scaling, transform features to lie in a given range, usually [0,1]. Z-score normalization or standard scaling is another method where features are scaled based on their mean and standard deviation. Properly scaled data ensures that each feature contributes equally to the model's decision, leading to more stable and accurate predictions.

- **Interaction Features[15]** refer to the creation of new features by combining two or more existing features, aiming to capture any synergistic effect between them. In many cases, the interaction between variables can provide more information than the individual variables themselves. For instance, while analyzing real estate prices, the individual features 'number of rooms' and 'location' might be informative, but their interaction, 'number of rooms in a specific location', might offer even more predictive power. Interaction features can be created by multiplying, adding, or even dividing original features, and they can help in capturing non-linear relationships in the data, enhancing the model's ability to make accurate predictions.

- **Feature Selection[15]** is a critical process in the machine learning pipeline that focuses on selecting the most relevant features from the original set, aiming to reduce the dimensionality and improve model performance. The primary goal is to eliminate redundant or irrelevant features that don't contribute significantly to the predictive power of the model. This not only helps in reducing the computational cost but also can lead to a more interpretable model. There are various techniques for feature selection, including filter methods (based on statistical measures), wrapper methods (like recursive feature elimination), and embedded methods (where algorithms inherently perform feature selection, such as decision trees). By judiciously selecting features, one can build efficient models that are less prone to overfitting and have better generalization capabilities.

5.3.2.1 Correlation tests

To assess feature quality, various statistical measures come into play, including correlation tests that gauge the strength and direction of relationships between variables. Pearson, Kendall, and Spearman correlation coefficients are frequently employed to quantify linear or monotonic associations between each feature and the target variable [3]. A high absolute value of these coefficients indicates a robust relationship, aiding in feature selection.

- **Pearson Correlation:** This measures linear relationships between two variables, ranging from -1 to 1. -1 signifies a strong negative linear correlation, 1 suggests a strong positive linear correlation, and 0 implies no linear correlation.

- **Kendall's Tau:** This non-parametric test gauges the strength and direction of a monotonic relationship between two variables.

- **Spearman's Rank:** Also non-parametric, it assesses how well an arbitrary monotonic function can describe the relationship between two variables without making assumptions about frequency distribution.

These techniques are valuable for evaluating relationships between each feature and generating correlation matrices, which, in turn, help identify redundant features. Univariate feature selection techniques allow the evaluation of each feature independently. In Python's scikit-learn library [26], methods like the F-test value and p-value are often used for this purpose.

- **F-test value:** This measures the linear dependency between the feature variable and the target. A higher F-test value suggests a more useful feature.
- **p-value:** It indicates the probability of an F-test value as large as observed arising if the null hypothesis is true. A smaller p-value implies rejecting the null hypothesis, indicating the feature's significance.

In summary, features constitute the foundational elements of any machine learning model. The quality of these features, their processing, and utilization significantly impact the model's performance. Feature engineering is of paramount importance, as properly engineered features can substantially reduce modeling errors, leading to more accurate and reliable predictions. It serves as a crucial link between raw data and predictive models, ensuring that models are fed with the most relevant and informative features.

5.3.2.2 Dimensionnality reduction

Following the aforementioned techniques, another essential facet in the feature engineering landscape is dimensionality reduction. As data grows in complexity, it often encompasses a vast number of features, leading to what is known as a high-dimensional space. While a plethora of features might seem advantageous, it introduces challenges, notably the *curse of dimensionality*[35, 17]. In such high-dimensional realms, data points tend to become increasingly sparse. This sparsity means that the relative distances between data points start to appear uniform, making it arduous for algorithms to discern meaningful patterns. This can lead to models that overfit the training data, capturing noise rather than the underlying data distribution. Additionally, the computational overhead increases, and deriving intuitive insights from the data becomes a daunting task.

Dimensionality reduction techniques come to the rescue by striving to trim down the number of features while preserving the crux of the information. Techniques like Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE) are employed to transform the data from its original high-dimensional space to a more manageable, lower-dimensional one. This transformation aims to retain the significant patterns and structures inherent in the data. By judiciously reducing the dimensionality, not only can models be trained more efficiently, but they often yield bet-

ter performance by focusing on the most pertinent features. This streamlined approach mitigates the challenges posed by the curse of dimensionality, ensuring models that are both robust and interpretable.

5.3.3 Imbalanced data

In machine learning, a frequent obstacle is the presence of datasets where one category vastly overshadows others[28]. This imbalance can skew models towards predicting the dominant class, often neglecting the less prevalent but potentially more critical class.

To counteract this, a variety of techniques have been devised:

- **Resampling:** This encompasses both increasing instances of the minority class (oversampling) and decreasing instances of the majority class (undersampling). A notable method for oversampling is the Synthetic Minority Over-sampling Technique (SMOTE), which generates artificial data points in the feature space.
- **Weighted Loss:** This strategy involves assigning greater weights to the minority class during the training phase, ensuring the model gives it due consideration.
- **Ensemble Methods:** Approaches such as bagging and boosting can be tailored to ensure a balanced class representation. For example, in bagging, each sample can be structured to maintain a balanced class ratio.
- **Anomaly Detection:** This method reframes the task from classification to anomaly detection, viewing the minority class as an outlier or anomaly.

Selecting an appropriate strategy hinges on the specific problem and dataset characteristics. It's also crucial to evaluate the model's efficacy using suitable metrics, ensuring it genuinely addresses the imbalance.

5.3.4 Some common models

- **Logistic Regression[25] :** Logistic regression serves as a statistical technique tailored for binary classification tasks. While linear regression is designed to forecast continuous outcomes, logistic regression focuses on predicting the likelihood of a binary result. It leverages the logistic function to relate multiple independent variables to a binary outcome, ensuring the predicted values fall between 0 and 1. Typically, a 0.5 threshold is used to classify the final outcome. A key strength of logistic regression is its clarity and ease of interpretation, though it might face challenges with complex non-linear data unless further feature adjustments are made.
- **Decision Trees[18] :** Decision trees are machine learning models designed for both classification and regression tasks. They segment data into subsets based on feature values, making decisions at each node. While their hierarchical structure offers easy visualization and interpretation, they

can be prone to overfitting. However, strategies like pruning can help in refining the tree and mitigating overfitting.

- **Random Forest[27] :** Random Forest is an ensemble method that creates a 'forest' of decision trees. Each tree is trained on a random subset of the data and makes its own predictions. The Random Forest algorithm then aggregates these predictions to produce a final result. This method is known for its high accuracy, ability to handle large datasets with higher dimensionality, and its capacity to manage missing values.
- **Support Vector Machines (SVM)[37] :** SVM are used for both regression and classification problems. They work by finding the hyperplane that best divides a dataset into classes. SVMs are effective in high-dimensional spaces and are versatile, as different Kernel functions can be specified for the decision function.
- **K-Nearest Neighbors (KNN)[19] :**KNN is a simple, instance-based learning algorithm. To make a prediction for a new data point, the algorithm finds the 'k' training examples that are closest to the point and returns the most common output value among them.

5.3.5 Random Forest classifier model

Random Forest, as introduced earlier, is an ensemble learning method that operates by constructing multiple decision trees during training and outputting the mode of the classes for classification tasks or mean prediction for regression tasks. Let's delve deeper into its workings, advantages, and disadvantages.

5.3.5.1 How Random Forest Works

1. **Bootstrap Aggregating (Bagging):** Random Forest begins by creating multiple datasets using bagging. It randomly selects samples from the original dataset with replacement, ensuring diversity in the datasets.
2. **Constructing Decision Trees:** For each of these datasets, a decision tree is constructed. However, instead of using all features to make a decision at a node, a random subset of features is chosen. This introduces further randomness into the model and ensures that the trees are uncorrelated.
3. **Aggregation:** Once all the trees are constructed, predictions are made by each tree. For classification, the mode of all the predictions is taken as the final prediction.

5.3.5.2 Advantages of Random Forest

- **Accuracy:** Random Forests often produce highly accurate predictions as they combine the output of multiple decision trees.
- **Overfitting:** The model is less prone to overfitting due to the randomness introduced in its construction.

- **Handling Missing Data:** Random Forest can handle missing values by either using median values to replace continuous variables or computing the proximity-weighted average of missing values.
- **Feature Importance:** It provides insights into the importance of different features in making predictions.

5.3.5.3 Disadvantages of Random Forest

- **Complexity:** Random Forest creates a lot of trees (unlike only one tree in the case of decision tree) and combines their outputs. This makes the model more complex and computationally intensive.
- **Longer Training Time:** Due to the construction of multiple trees, the training time can be longer compared to other algorithms.
- **Less Intuitive:** While individual decision trees are simple and can be visualized, a forest is more challenging to interpret due to its ensemble nature.

In conclusion, while Random Forest offers high accuracy and is robust against overfitting, it comes with increased complexity and training time. However, its advantages often outweigh the disadvantages, making it a popular choice for various machine learning tasks.

5.4 Clustering

Clustering is a key technique in unsupervised machine learning, aiming to group similar data points together. It's all about ensuring that items within a cluster are more alike than those in different clusters. This approach is great for uncovering hidden patterns in data. When it comes to checking the quality of an embedding, clustering can be a handy tool. By forming clusters from embedded data, we can see how effectively similar structures come together. A top-notch embedding should make sure that data points from the same structure cluster closely. So, by looking at how well clustering works, we can gauge the strength of the embedding.

5.4.1 K-Means Clustering

K-Means [23] is a popular clustering method recognized for its straightforwardness and speed. It works by dividing a dataset into 'K' unique, separate groups (or clusters) based on how close each data point is to the cluster's center, termed the centroid. The method repeatedly places each data point with the closest centroid and then updates the centroid's position based on the points in its cluster. This cycle repeats until the centroids no longer move significantly. Though K-Means is great for clusters that are roughly spherical and similar in size, deciding on the best number of clusters (K) in advance can be tricky.

5.4.2 DBSCAN

DBSCAN [8] is a clustering method that identifies dense regions in data, considering sparse areas as outliers. Unlike K-Means, DBSCAN doesn't need a pre-defined number of clusters. It works on the principle that a cluster is a high-density area in data, surrounded by less dense regions. The algorithm is steered by two key parameters: the minimum points needed for a region to be dense and a distance measure determining how close points should be to create a cluster. DBSCAN shines in handling datasets where clusters have varying shapes but similar densities.

5.4.3 Spectral Clustering

Spectral clustering[22] is a method that emphasizes reducing the dimensionality of data using the eigenvalues of a similarity matrix. By constructing a similarity graph, where nodes represent data points and edges carry weights based on point similarities, the technique transforms the original space. Utilizing the eigenvectors of the graph's Laplacian, it creates a more compact and manageable representation. In this reduced space, traditional clustering methods like K-Means become more effective. Spectral clustering's strength lies in its ability to handle complex cluster structures, especially non-convex clusters, making it a valuable tool for diverse applications.

5.4.4 OPTICS Clustering

Ordering Points To Identify the Clustering Structure (OPTICS) [1] is an advanced density-based clustering algorithm that was introduced as an extension to DBSCAN. The primary motivation behind OPTICS was to address some of the limitations of DBSCAN, especially its sensitivity to the global density parameter. Let's dive deeper into the workings, advantages, and disadvantages of OPTICS.

5.4.4.1 How OPTICS Works

1. **Ordering of Data Points:** OPTICS begins by processing each data point in the dataset. For each point, it computes its reachability distance with respect to other points. This distance is based on the density of the data, and it gives an indication of how close or far a point is from a dense region.

2. **Reachability Plot:** The reachability distances for all points are then used to create a reachability plot. This plot provides a visualization of the density-based clustering structure of the data. Peaks in the plot represent sparse regions, while valleys correspond to dense clusters.

3. **Cluster Extraction:** Clusters can be extracted from the reachability plot by analyzing its valleys. Points within a valley belong to the same cluster. The depth and shape of a valley give insights into the density and shape of the cluster.

5.4.4.2 Advantages of OPTICS

- **Varying Densities:** Unlike DBSCAN, which struggles with clusters of varying densities, OPTICS can identify clusters that have different density levels within the same dataset.
- **No Global Density Parameter:** OPTICS does not require a global density threshold, making it more adaptive to the data's inherent structure.
- **Visualization:** The reachability plot provides a visual representation of the clustering structure, aiding in the interpretation of results.

5.4.4.3 Disadvantages of OPTICS

- **Complexity:** OPTICS is computationally more intensive than DBSCAN due to the ordering of data points and the generation of the reachability plot.
- **Interpretation:** While the reachability plot provides a visual representation, extracting clusters from it can be challenging and may require additional heuristics or methods.

5.4.4.4 Parameters of OPTICS

The OPTICS algorithm also provides a range of parameter choices to cater to different clustering needs:

- **clusterization method:**
  - `"xi"`: The ξ method, often referred to as the "steepness" method, is designed to extract clusters from the reachability plot generated by the OPTICS algorithm. The reachability plot is a visualization where data points are plotted based on their reachability distances. In this plot, clusters manifest as valleys, and the depth or steepness of these valleys indicates the density of the cluster. The ξ method works by:
    * Identifying regions in the reachability plot where there is a significant change in the reachability distance, indicating potential cluster boundaries.
    * Determining the "steepness" of these regions. A region is considered steep if the relative change in reachability distance exceeds a threshold, typically denoted by the parameter ξ.
    * Extracting clusters based on these steep regions. Clusters are formed by connecting steep upward regions (representing the start of a cluster) with steep downward regions (indicating the end of a cluster).
    * Handling noise: Points that do not belong to any of the identified steep regions are typically considered as noise or outliers.

    The advantage of the ξ method is its ability to detect clusters of varying densities without the need for specifying the number of clusters in advance. However, the choice of the ξ

parameter can influence the granularity of the clustering, with smaller values leading to more fine-grained clusters and larger values resulting in coarser clusters.

  - Other methods: Include the DBSCAN-equivalent method and others, each offering a unique approach to cluster extraction.

- **clusterization metrics:**

  - `"cosine"`: Cosine similarity is a metric used to determine the cosine of the angle between two non-zero vectors in an inner product space. It is especially suitable for high-dimensional datasets, such as text data represented as vector space models or TF-IDF vectors. The cosine similarity, $\text{sim}(A, B)$, between two vectors A and B is given by:

$$\text{sim}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} \tag{5.8}$$

    where:

    * θ is the angle between vectors A and B.
    * $A \cdot B$ represents the dot product of the vectors.
    * $\|A\|$ and $\|B\|$ denote the magnitudes (or norms) of vectors A and B, respectively.
    * **Advantages**:
      · *Scalability*: Cosine similarity is less affected by the magnitude of vectors, making it advantageous for data that doesn't scale linearly. This means that even if the data is not normalized, cosine similarity can still provide meaningful results.
      · *High-dimensional data*: In high-dimensional spaces, Euclidean distances can become inflated, a phenomenon known as the "curse of dimensionality." Cosine similarity, on the other hand, remains robust in such scenarios, making it a preferred choice for datasets with many features.
      · *Sparse data*: For datasets where vectors are sparse (i.e., have many zeros), cosine similarity can be more efficient and meaningful than other distance metrics. This is because it focuses on the orientation (angle) between vectors rather than their absolute differences.

  - Other metrics: Include Euclidean, Manhattan, Jaccard, and more, each with its distinct advantages.

- **clusterization algorithm:**

  - `"brute"`: Employs the brute-force approach to compute pairwise distances, ensuring exact distance calculations (allow cosine metrics).
  - Other algorithms: Such as "kd-tree" or "ball-tree", which can offer faster computations for specific datasets.

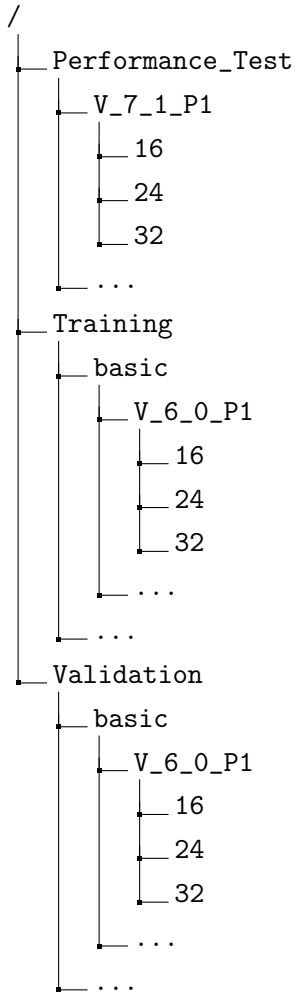
In conclusion, OPTICS offers a more flexible approach to density-based clustering compared to DBSCAN. Its ability to handle clusters of varying densities and its adaptive nature make it a powerful tool for clustering tasks. However, its increased computational complexity and challenges in cluster extraction require careful consideration.

5.5 Dataset

SmartKex has enriched the research domain by curating an extensive annotated dataset of OpenSSH heap memory dumps, which is publicly accessible on Zenodo[1] [9].

The dataset is systematically structured into three primary directories: *Training*, *Validation*, and *Performance_Test*. Both the *Training* and *Validation* directories are further segmented based on distinct SSH scenarios, including immediate exit, port-forward, secure copy, and shared connection. Each scenario directory is then subdivided according to the OpenSSH version that produced the memory dump. Within these version-specific directories, heap dumps are organized by their respective key lengths, offering a hierarchical structure that facilitates targeted research explorations.

Figure 5.1: Schematic Representation of the Dataset's Directory Hierarchy



The dataset predominantly employs two file formats: JSON and RAW. While the JSON files encapsulate meta-data such as the encryption technique, the virtual memory address of the key, and its hexadecimal representation, the RAW files house the actual memory dump of the OpenSSH process.

[1] `https://zenodo.org/record/6537904`

5.5.1 Details of the Dataset Production System

While the [9] paper and the associated dataset do not explicitly detail the hardware and software configurations used during its creation, such information is pivotal. This is especially true as our analysis delves into allocated raw bytes, which are influenced by the underlying system and the C library in use. To bridge this information gap, we reached out to the authors directly.

In correspondence with Dr. Hans Reiser, we were provided with specifics about the system used to generate the dataset. The system's configuration was ascertained using the following command outputs:

```
root@debian10:~# ldd --version
ldd (Debian GLIBC 2.28-10) 2.28
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
     There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A
   PARTICULAR PURPOSE.
Written by Roland McGrath and Ulrich Drepper.
```

Code 5.1: C-library version utilized during dataset generation

```
root@debian10:~# lsb_release -a
No LSB modules are available.
Distributor ID:     Debian
Description:        Debian GNU/Linux 10 (buster)
Release:            10
Codename:           buster
```

Code 5.2: Linux Standard Base Release details

```
root@debian10:~# uname -a
Linux debian10 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1
   (2021-03-19) x86_64 GNU/Linux
```

Code 5.3: Operating system and kernel version details

Dr. Reiser further mentioned that the dataset was produced on a system powered by an Intel Xeon CPU, specifically either the E5-2609 or the E3-1230 model. Based on the provided commands, we can summarize the system's key attributes as:

- **CPU architecture:** `x86_64`
- **Operating System:** `Debian GNU/Linux 10 (buster)`
- **Kernel version:** `4.19.0-16-amd64`
- **C library version:** `Debian GLIBC 2.28-10`

5.5.2 C structures and chunks allocation understanding

Given that the dataset encompasses entire heap dump files, it presents an opportunity to identify potential data structures within these dumps. Identifying these structures involves searching for patterns within the heap dump. However, when it comes to data structures, our insights into the C library in use become invaluable, guiding our search.

OpenSSH, being crafted in C, naturally embeds C data structures within its heap dumps. In C, memory allocation for data structures is typically handled by the `malloc` function. This function, when invoked, allocates memory corresponding to the size of the specified data structure and returns a pointer to this allocated space (also call chunks). Given our knowledge that the dataset was generated with `GLIBC 2.28` 5.5.1, a dive into the `malloc` function within `GLIBC 2.28` reveals a noteworthy detail. The comments within the code mention that each allocated chunk carries a minimal overhead, either 4 or 8 bytes, which holds size and status details [11]. This overhead is what we term the *malloc header*. Consequently, we can anticipate the presence of 8-byte aligned blocks within the heap dump that aren't pointers but are remnants of a `malloc` invocation. Recognizing these *malloc headers* paves the way for detecting potential data structures within the heap dumps.

On a Linux system with a `x86_64` architecture, the `malloc` function typically employs a block (or chunk) header to store metadata about each allocated segment. Positioned right before the memory block returned to the user, the specifics of this header can vary based on the C library's implementation (e.g., glibc, musl). However, it generally encapsulates:

- **Size of the Block**: This represents the allocated block's size, typically denoted in bytes. Often, this size encompasses the header's size and might be aligned to multiples of 8 or 16 bytes.
- **Flags**: These are a set of indicators that shed light on the block's status. They can signify if the block is free or allocated, or even if the preceding block is free or allocated. Ingeniously, these flags are often stashed in the size field's least significant bits, leveraging the alignment of the size which zeroes out these bits.

5.5.3 Understanding `malloc` in Heap Memory Allocation

The `malloc` function, as implemented in GLIBC 2.28, employs a boundary tag methodology to oversee memory chunks. Each of these chunks incorporates metadata essential for both memory allocation and deallocation [11] [6].

A chunk represents a continuous memory segment, typically comprising multiple 8-byte blocks, managed by `malloc`. The structure of a chunk encompasses the following elements [6] [33]:

1. **Size of Previous chunk**: Present only if the preceding chunk is free (with its `P` (PREV_INUSE) bit unset), this field assists in locating the start of the prior chunk.

2. **Size of chunk**: This captures the chunk's byte size and integrates three flags: `A` (NON_MAIN_ARENA), `M` (IS_MMAPPED), and `P` (PREV_INUSE). These flags reside in the size field's last three LSBs. This block is often referred to as the *malloc header* block in subsequent discussions.

3. **User Data**: The actual memory segment returned for user utilization (it will be minimum 2 blocks, 16 bytes).

4. **Size of Next chunk**: Represents the size of the succeeding contiguous chunk.

5. **Foot**: Mirrors the chunk's size and is reserved for application data.

6. **Flags**:

    - `A` (NON_MAIN_ARENA): Denotes if the chunk resides in the primary or a thread-specific arena.
    - `M` (IS_MMAPPED): Flags if the chunk was allocated via `mmap`.
    - `P` (PREV_INUSE): Signifies if the preceding chunk is occupied. If unset, the prior chunk is free.

The chunk allocation mechanism is underpinned by:

1. **Initialization**: The inaugural chunk allocated invariably has its `P` bit activated to avert accessing non-existent memory.

2. **Free chunks**: These chunks are cataloged in circular doubly-linked lists, encompassing forward and backward pointers for navigation.

3. **Mmapped chunks**: Such chunks, identifiable by the `M` bit in their size fields, are allocated individually.

4. **Fastbins**: Regarded as allocated chunks, their consolidation is executed collectively.

5. **Top chunk**: A unique chunk that perpetually exists. If it dwindles below `MINSIZE` bytes, it undergoes replenishment.

The code comments offer a detailed chunk representation, comprising 8-byte blocks [11]. It's worth noting that this representation is tailored to align with our forensic analysis needs. A subtle distinction lies in the `footer`'s portrayal, which, for our purposes, is deemed part of the succeeding chunk rather than the current one. The footer of the prior chunk essentially corresponds to the `mchunkptr` address. As elucidated in the GlicC wiki, a "chunk pointer" or `mchunkptr` doesn't point to the chunk's commencement but to the terminal word in the preceding chunk. This implies that the initial field in `mchunkptr` is valid only when the prior chunk is free [6]. Given the interplay between free and allocated chunks, it's more intuitive to perceive the footer as an element of the next chunk. This schematic representation divergence doesn't alter the actual data structure but merely offers a clearer visualization.

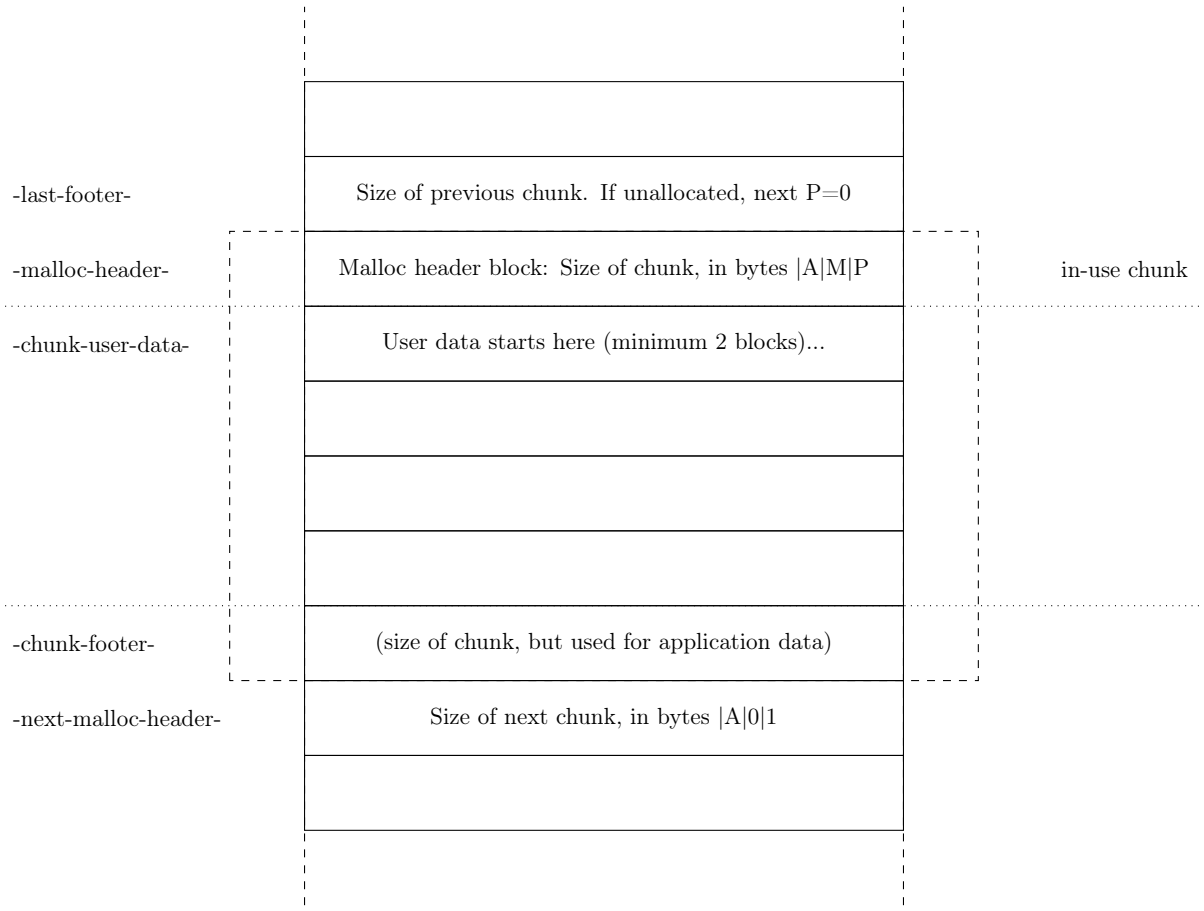


Figure 5.2: Diagram of an allocated chunk in GLIBC 2.28 [11].

In GLIBC 2.28, the `malloc` function employs a boundary tag approach to oversee memory chunks. These chunks integrate metadata essential for memory allocation and deallocation [11] [6]. The library organizes available chunks into circular doubly-linked lists, termed "bins", facilitating rapid retrieval of free memory chunks of specific sizes. However, these bins are not directly accessible in the heap dump file. To discern if a particular chunk is occupied or available, several techniques can be employed. Primarily, the `P` bit in the malloc header serves as an indicator. A value of 1 denotes an occupied chunk, while 0 signifies a free chunk.

It's noteworthy that certain heap dump files appear truncated, with the concluding block being incomplete and filled with zeros. An instance of this can be observed in the final chunk of *Training/basic/V_7_1_P1/24/17016-1643962152-heap.raw*.

```
1 WARN: chunk [94022266975200] Chunk(block_index=10876, size
     =48176, flags=[A=False, M=False, P=True]) is out of bounds.
     Last block index: 16895 Iteration index: 16896
2 WARN: chunk [94022266975200] Chunk(block_index=10876, size
     =48176, flags=[A=False, M=False, P=True]) is out of bounds.
     Last block index: 16895 Iteration index: 16897
3 Chunk(block_index=10876, size=48176) is only composed of zeros.
```

Code 5.4: Logs from chunk exploration script, highlighting the last chunk of the file *Training/basic/V_7_1_P1/24/17016-1643962152-heap.raw*.

A free chunk, as per the code documentation [11], incorporates pointers to the subsequent and preceding free chunks within the heap for its designated bin. The following provides a representation of a free chunk:

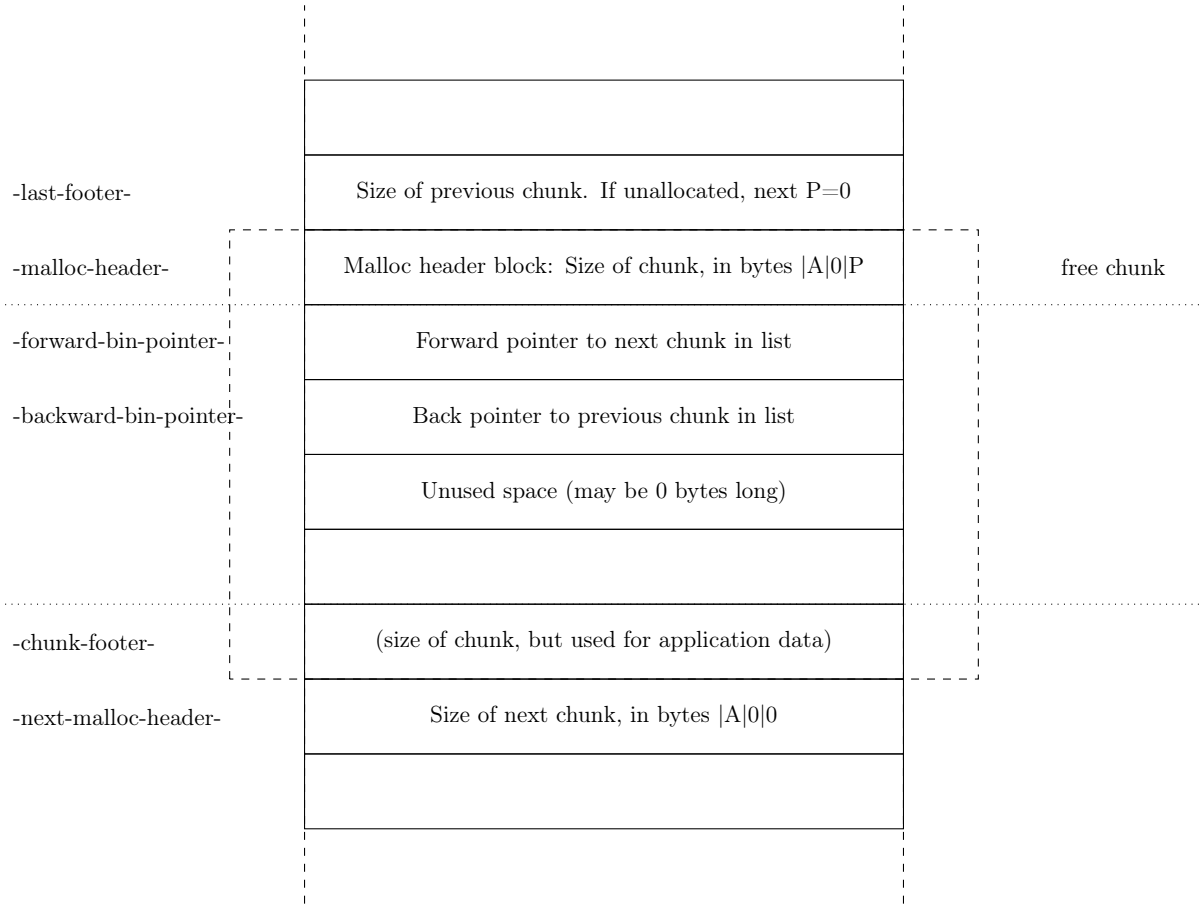


Figure 5.3: Diagram of a free chunk in GLIBC 2.28 [11].

The principle of **chunk chaining** is pivotal for navigating the heap dump file. By adhering to the sequence of malloc headers, we can systematically traverse the allocated memory chunks within the heap dump. This methodology is corroborated by the source code, which contains a comment indicating that, given the address of the initial chunk (the one with the lowest address) in a heap, one can iterate over all the chunks by leveraging the size data.

Throughout the development of scripts and tools for this thesis, we've integrated various checks and validations to ensure the consistency of this chunk chaining assumption. Should any discrepancies arise, the tools are designed to flag an error, and typically bypass the problematic data. Such a design choice aims to fortify the tools against unforeseen data structures and to bolster the reliability of the outcomes.

6 Methods

This research dives into the complexities of embedding byte sequences, focusing particularly on the extraction of structures containing SSH keys for machine learning purposes. The varied uses of OpenSSH introduce distinct challenges due to potential variations in the created embeddings. Given the wide array of SSH key dimensions and OpenSSH's intricate operations, maintaining the embeddings' stability and consistency is vital. In this methodological section, we will detail various embedding methods, present a framework for their assessment through a classifier model, and suggest another strategy to verify the embeddings' coherence between the different OpenSSH usage and key sizes.

6.1 Dataset

The dataset at the core of this thesis, as previously introduced (see 5.5), consists of heap dump raw files related to different OpenSSH use cases and versions. Each heap dump file is paired with a JSON annotation file created by the dataset's creators. These JSON files provide extra information about the heap dump, especially regarding encryption keys. In this section, we will explain our exploration of the dataset, aiming to better comprehend its content and nuances.

6.1.1 Origin

The dataset is derived from heap dumps that capture various OpenSSH usage scenarios. These scenarios encompass four distinct SSH interactions: a straightforward client connection to the server followed by an immediate exit, port-forwarding, secure copying, and SSH shared connection. The heap dumps span different OpenSSH versions and a range of key sizes, from 16 to 64 bytes. These dumps were generated using the SmartKex tool [9]. The data collection was conducted on a mini PC equipped with an AMD Ryzen 5500U processor, 16GB of RAM, and a 1TB NVMe SSD, running Debian 11 as its operating system.

6.1.2 Estimating the dataset balancing for key prediction

In this part, our primary objective was to assess the balance of the dataset for key prediction and identify the challenges associated with it.

To begin, we aimed to gain an understanding of the dataset's scale. We utilized a code snippet 6.1 to count all the files within the dataset, revealing a total of 208,745 files. However, it was imperative to recognize that JSON files, which served as annotation files, were not to be considered part of the raw bytes for embedding. Consequently, these JSON files were excluded from our count to provide a more accurate representation of the dataset's size.

```
1         find . -type f | wc -l
```

Code 6.1: Count all dataset files

Following this, we employed another code snippet 6.2 to specifically count the heap dump raw files, excluding JSON files. This count indicated a total of 103,595 heap dump raw files, which constituted the primary focus of our analysis.

```
1         find . -type f -name "*.raw" | wc -l
```

Code 6.2: Count heap dump raw dataset files

To gain further insights into the dataset, we determined its size while excluding annotation files 6.3. The calculated dataset size amounted to 18,067,001,344 bytes.

```
find . -type f -name "*.raw" -exec du -b {} + | awk '{s+=$1} END {
    print s}'
```

Code 6.3: Get the size of the dataset

Considering the nature of the dataset, which featured a maximum of six keys per file, each with a maximum size of 64 bytes, we conducted a rough estimate. We determined that the maximum number of bytes relevant for searching across the dataset was $6 * 64 * 103595 = 39780480$. This calculation accounted for approximately 0.22% of the dataset's total size.

Lastly, it is crucial to acknowledge that the dataset exhibited a significant imbalance and is very large. To address this challenge effectively, strategies were implemented to ensure robust, unbiased analyses, and scalability.

Annotations

The annotations files are essential to understand the data and how best to utilize them for the study. Each heap dump corresponds to one specific JSON file. To view the contents of these JSON files in a more organized manner, one can reference the method provided at 6.4. For a clearer understanding, an extract of the JSON annotation from the file located at `./Training/client/V_7_8_P1/16/13116-1644920217.json` is available at 6.5.

```
python3 -m json.tool file.json
```

Code 6.4: pretty print JSON

```
{
    /* file ./Training/client/V_7_8_P1/16/13116-1644920217.json
       */
    "SSH_STRUCT_ADDR": "5619dd7e5570",
    "SESSION_STATE_ADDR": "5619dd7e5df0",
    "KEY_A_ADDR": "5619dd807f40",
    "KEY_A_LEN": "12",
    "KEY_A_REAL_LEN": "12",
    "KEY_A": "34fbe182e76c49a617a93e2e",
    /*...*/
    "KEY_E_ADDR": "5619dd808000",
    "KEY_E_LEN": "0",
    "KEY_E_REAL_LEN": "0",
    "KEY_E": "",
    "KEY_F_ADDR": "5619dd807fd0",
    "KEY_F_LEN": "0",
    "KEY_F_REAL_LEN": "0",
/    "KEY_F": "",
    "HEAP_START": "5619dd7e3000"
}
```

Code 6.5: An extract of the JSON annotations

Within these annotation files, several critical pieces of information are present. The "SSH_STRUCT_ADDR" and "SESSION_STATE_ADDR" denote the addresses of vital openSSH structures. These addresses are pivotal in gauging the embedding coherence across different openSSH uses and key sizes. If the embeddings of these structures display similarity across various key sizes and openSSH usages, it signifies the embedding's coherence.

Other significant annotations such as "KEY_A_ADDR", "KEY_A_LEN", "KEY_A_REAL_LEN", and "KEY_A" detail the address, length, and value of the key A. In general, six of these annotations can be found for each heap dump. Notably, the "HEAP_START" annotation, along with the length of the heap dump, is of paramount importance. This annotation signifies the starting address of the heap dump. This information not only aids in pinpointing addresses in the heap dump for structures and pointers, but also refines the heuristic used in detecting pointers. By leveraging the "HEAP_START" information, one can verify if a pointer is pointing within the heap dump boundaries. As a practical illustration, deducing the address of key A within the heap dump can be achieved by subtracting "HEAP_START" from "KEY_A_ADDR".

However, it's noteworthy that some of these annotation files may be corrupted. Therefore, it's imperative to verify the integrity of each file before its use. In instances where keys are corrupted, such as "KEY_E" and "KEY_F" having no recorded values in the extract found at 6.5, it's advised either to remove the corrupted keys or discard the entire file if the data cannot be salvaged.

6.1.3 Dataset Validation

The dataset primarily consists of heap dump RAW files, each corresponding to various use cases and versions of OpenSSH. Accompanying each heap dump is a JSON annotation file, crafted by the dataset's creators, to furnish supplementary details, particularly about encryption keys.

However, the dataset isn't without its flaws. Its application in machine learning has unveiled certain inconsistencies. For example, a few of these files are incomplete, lacking essential data. This poses a challenge since we rely on these annotations to pinpoint key addresses, crucial for annotating memory graphs in the embedding phase. If there's a discrepancy in the format, we'll deem the JSON annotation as corrupted and bypass it. This likely stems from the automated generation of annotations. A case in point is the file in *Training/basic/V_7_8_P1/16/*, which, being the dataset's first file, showcases an incomplete annotation with absent keys. It's vital to be cognizant of these limitations when utilizing the dataset for academic endeavors.

6.1.3.1 Annotation Integrity Verification

To accurately gauge the usability of the dataset for machine learning applications, we implemented a validation script named `check_annotations.py`. This script is tailored to assess the annotations for their quality, completeness, and consistency.

The annotations (JSON files) are categorized as follows:

- **Complete and Accurate Files**: These files are devoid of missing keys and contain all keys with appropriate values.
- **Malformed Files**: These are files that aren't valid JSON and hence cannot be loaded properly.
- **Inconsistent Files**: Files that present conflicting information within their annotations.
- **Files with Absent Keys**: These files lack certain keys in their annotations. For instance, a JSON file might have "KEY_E": "", indicating the absence of key E and its corresponding address in the annotation, which poses challenges for accurate machine learning labeling.
- **Files with Incomplete Keys**: These files contain keys but lack the corresponding addresses. An example would be a JSON file with "KEY_E": "689e549a80ce4be95d8b742e36a229bf", signifying the presence of key E but the absence of its address in the annotation. This again complicates the labeling process for machine learning.

The script executes swiftly, processing all the 103,595 JSON annotation files and yields the following outcomes:

- **Correct and Complete Files**: 26,202 files.

- **Broken Files**: 6 files are identified as broken. Closer inspection reveals these files to be empty.
- **Incorrect Files**: 0 files.
- **Files with Absent Keys**: 58,643 files exhibit missing keys.
- **Files with Incomplete Keys**: 18,750 files display incomplete keys.

Delving deeper into the keys:

- **Total SSH Keys**: 546,534 keys.
- **Missing (Empty) SSH Keys**: 157,244 keys.
- **Incompletely Annotated SSH Keys**: 37,500 keys.
- **Incorrectly Annotated SSH Keys**: 0 keys.

6.1.4 Structure of the Heap File

Heap files serve as the dynamic memory storage for applications, and understanding their structure is crucial for memory analysis. These files are organized in a specific manner, with memory sequences of bytes or "chunks" allocated and deallocated as needed by the application. The heap is 8-byte aligned, which means that we can consider sequences of memory in 8-byte block. This alignment ensures efficient memory access and management. To visualize and interpret the heap's structure, tools like memory analyzers or debuggers can be employed.

Within the heap, there are four primary types of byte sequences that can be identified with varying degrees of certainty:

1. **Chunk, Malloc Header, and Footer**: We have already discussed these components in detail in an earlier section. In brief, they represent the primary building blocks of the heap, with each chunk being a segment of memory allocated for storing data. The malloc header contains essential metadata about the chunk, and footers, when present, replicate this information.
2. **Pointer**: Memory addresses that reference other locations within the heap or other memory segments.

Any unidentified user data within these structures is termed as "value data." This data represents the actual content or payload stored within the allocated memory chunks.

6.1.4.1 Chunk

In our exploration of the dataset, the chunk chaining assumption, as detailed in section 5.5.3, plays a pivotal role. It's imperative to ensure the integrity of this assumption for accurate analysis. During the dataset refinement process, we identified five heap dumps that contain chunks with a size of 0 bytes, which could potentially violate this assumption. To maintain the reliability of our analyses, these specific dumps have been removed from the dataset.

6.1.4.2 Pointer

Pointers are memory addresses that reference other locations within the heap or other segments of memory. In the context of the heap, pointers can indicate data structures, reference other chunks, or provide links in data structures like linked lists or trees. To identify potential pointers within the heap dump, one can utilize the following Reggex :

```
1 :/[0-9a-f]\{12}0\{4}
```

This command searches for sequences comprising 12 hexadecimal characters succeeded by 4 zeros. The rationale behind this is twofold:

- The heap dump file's maximum possible addresses typically span around 12 hexadecimal digits.
- Pointers' addresses are represented in little-endian format. Consequently, the address's last 4 bytes at 0 are its Most Significant Bytes (MSB).

Furthermore, with knowledge of the heap's start addresses and the dump's size, we can enhance the precision of our search. By doing so, we can exclude potential pointers that point outside the boundaries of the dump. Another refinement can be made by verifying if the values pointed to by the potential pointers are 8 bytes aligned, as the heap is structured in 8-byte sequences. However, it's crucial to note that this approach remains heuristic in nature. As such, there's still a possibility of detecting blocks that aren't genuine pointers.

6.1.4.3 Footer

In the GLIBC documentation, the footer of a chunk is expected to mirror the chunk's size as indicated in the malloc header. However, an inconsistency is observed: the size stated in the footer block doesn't always align with this expectation. This deviation is consistently seen across the refined dataset.

6.1.5 Heap File Distribution

The dataset offers an in-depth perspective on heap dumps, systematically sorted by key size, OpenSSH version, and distinct use cases. This methodical arrangement streamlines the analytical

process, enabling precise investigations tailored to particular criteria. In the subsequent sections, we'll delve into the distribution of the dataset, emphasizing the file count across different use cases, versions, and key sizes, to ensure its suitability for consistent testing.

6.1.5.1 Full Dataset

In our initial phase of exploration, we will concentrate on the full dataset. This comprehensive analysis will provide a holistic understanding of the data's structure, variations, and potential anomalies.

- A unique instance was observed where a folder was devoid of any content. This was in the Training section, specifically for the client use case, version V_7_8_P1, and a key size of 64 bytes.
- In the Training segment, which comprises 82 combinations of use cases, versions, and key sizes:
  - The minimum number of RAW files present is 923.
  - The maximum stretches to 1095. The difference between these two extremes is calculated as:
    $$\frac{\max - \min}{\min} = 0.186 \tag{6.1}$$
    This results in a 18.6% difference.
- For the Testing segment, which has 15 combinations:
  - The RAW files range from a minimum of 100 to a maximum of 101, marking a mere 1% difference between the two.
- The Validation segment, with its 82 combinations, shows:
  - A minimum of 151 RAW files.
  - A maximum of 211 RAW files. The difference between these values is:
    $$\frac{\max - \min}{\min} = 0.397 \tag{6.2}$$
    This presents a 39.7% difference, yet the number of files remains substantial enough to validate any model effectively.

The dataset, with its meticulous organization and vast range, offers a robust platform for in-depth analysis and model validation.

6.1.5.2 Clean Dataset

Following our examination of the full dataset, we will shift our focus to the cleaned dataset. This refined subset, having undergone meticulous preprocessing and filtering, will offer insights into the most pertinent and reliable data points. Analyzing the cleaned dataset will ensure that our conclusions and subsequent actions are based on high-quality, accurate data.

- In the Training segment, which comprises 82 combinations of use cases, versions, and key sizes:
  - 63 subdirectories are empty, with no RAW files present.
  - The minimum number of RAW files present is 923.
  - The maximum stretches to 1079. The difference between these two extremes is calculated as:
    $$\frac{\max - \min}{\min} = 0.169 \tag{6.3}$$
    This results in a 16.9% difference.
- For the Testing segment, which had 15 combinations : 0 subdirectories are empty, then no changes are observed.
- The Validation segment, with its 82 combinations, shows:
  - 62 subdirectories are empty, with no RAW files present.
  - A minimum of 151 RAW files.
  - A maximum of 209 RAW files. The difference between these values is:
    $$\frac{\max - \min}{\min} = 0.384 \tag{6.4}$$
    This presents a 38.4% difference, yet the number of files remains substantial enough to validate any model effectively.

The specifics of the empty folders, including their exact locations and other details, will be cataloged comprehensively in the annex B.1. It's crucial to note that due to the invalid nature of the data in these folders, our coherence study on the embeddings will not factor in the OpenSSH version, use case, or key size involved. This decision ensures that our analysis remains rooted in valid and meaningful data, thereby enhancing the reliability of our findings.

While the cleaning process did invalidate certain cases within the dataset, it's essential to emphasize that a significant portion remains intact and consequential. These preserved cases provide a robust foundation for our analysis, ensuring that our study is both comprehensive and grounded in meaningful data. The invalidated cases, though notable, do not diminish the overall value and depth of the dataset at our disposal.

6.1.6 Keys Analysis

The analysis of SSH keys within the heap dumps provides crucial insights into their characteristics and behaviors. These findings not only enhance our understanding of the data but also guide subsequent steps in the research process.

6.1.6.1 Keys Positions

Upon analyzing all the heap dumps, it became evident that all the SSH keys mentioned in the annotations are positioned at the beginning of their respective chunks. These keys have a size ranging from a minimum of 12 bytes to a maximum of 64 bytes. Additionally, the size of the chunks in which these keys are found is consistently observed to be 32, 48, or 64 bytes. This consistent positioning and size uniformity greatly simplify certain embedding processes and offer a streamlined approach to further analysis.

6.1.6.2 Keys Entropy

Another significant observation regarding the keys is their high entropy, as referenced in 4.2. High entropy is indicative of randomness, which is a characteristic feature of cryptographic keys. Leveraging this high entropy 5.1 can be instrumental in discriminating the keys from other data. However, it's essential to approach this method with caution. While high entropy can be a strong indicator, it's not foolproof. There's a possibility of encountering false positives, as other high entropy data might exist in the heap. Additionally, there might be instances of false negatives, especially when keys contain multiple repeated bytes by chance.

6.2 Embedding

Our next objective centers on the conversion of raw byte data into fixed-size embeddings (5.1, 5.2), a pivotal step in preparing them for utilization in machine learning applications. Ensuring uniformity in embedding size across all memory structures holds paramount significance. Consistency in embedding dimensions is vital to empower machine learning algorithms for efficient data processing and analysis. This uniformity not only simplifies the integration of memory structures with varying sizes into a coherent classification framework but also acts as a defense against the adverse effects of the curse of dimensionality—a phenomenon that can introduce computational complexities and heighten the risk of overfitting in high-dimensional data spaces. Striking this equilibrium is essential, achieved by maintaining reasonably low embedding dimensions, fostering both efficient data processing and the preservation of essential information within the raw byte data. It's important to note that initially, each embedding will include the structure's file and the structure's address in the file. However, these details will be removed during the machine learning phase (quality or coherence) as the embedding aims to be free of key size or OpenSSH uses. Their presence will serve as a means to test coherence later in our analysis.

6.2.1 First Preprocessing

Initially, we possess some knowledge regarding the positions and characteristics of SSH keys. Leveraging this knowledge, we can narrow down the number of chunks to examine, thereby streamlining the analysis process.

6.2.1.1 Entropy-Based Approach

A distinctive feature of SSH keys is their inherent randomness, as referenced in section 5.1. This randomness translates to a high entropy value for the keys. By focusing on chunks with high entropy, we can potentially isolate those that contain SSH keys.

To compute the entropy, we consider the minimal size of a key, which is 12 bytes as mentioned in section 6.1.6. Given that a chunk has a minimum of 2 blocks, equivalent to 16 bytes as detailed in section 5.5.2, and the fact that keys are always positioned at the beginning of a chunk, we can extract the first 12 bytes of a chunk to compute its entropy.

Given the abundance of data at our disposal, it's feasible to focus solely on chunks with the highest entropy values. However, it's crucial to remember that this is a heuristic approach. There's a need to be vigilant and ensure that important nodes with slightly lower entropy, possibly due to random occurrences like two identical bytes in the initial 12 bytes, are not inadvertently excluded from the analysis. Due to the potential risk of missing out on some keys with this filter, it is advisable to apply it only during the training phase and not in the validation phase. However, for the sake of simplicity

in our approach, we will apply this filter in the validation phase as well, taking extra care to add any missing keys back into the dataset to ensure completeness and accuracy in our analysis.

6.2.1.2 Chunk Size Filter

Our analysis of SSH key chunks has shown that they are always either 32, 48, or 64 bytes in size. This observation allows us to apply a filter based on chunk size, selecting only chunks of these specific sizes for further analysis and discarding the rest. This method of selection helps to streamline the analysis process and ensures that we are focusing on the most relevant chunks.

By combining the entropy-based approach and the chunk size filter, we can avoid the need for random subsampling. This is particularly advantageous as our dataset is not significantly imbalanced, provided that these filters are applied.

6.2.2 Basic Chunk Information

Every chunk in the heap dump is equipped with a set of fundamental details that provide insights into its structure and content. These basic pieces of information are essential for understanding the layout and composition of each chunk.

It's important to note that these foundational details are not exclusive to just the primary chunk nodes. Value nodes and pointer nodes, which might be considered as sub-components of a chunk, also inherit these basic attributes from their parent chunk node.

The core information associated with each chunk includes:

- **block_position_in_chunk**: This represents the position of a specific block within the chunk. For the primary chunk node, this value is always 0, indicating the start of the chunk.
- **chunk_byte_size**: This provides the total size of the chunk, measured in bytes.
- **chunk_ptrs**: This denotes the total number of pointers present within the chunk.
- **chunk_vns**: This indicates the total number of value nodes contained within the chunk.
- **chunk_number_in_heap**: This value represents the index or position of the chunk within the entire heap, giving a relative placement of the chunk in the context of all chunks.

With these basic details, one can gain a comprehensive understanding of each chunk's structure, content, and relative position within the heap.

6.2.3 Statistical embedding

Understanding the fundamental concepts of statistical embeddings enables us to delve deeper into the sophisticated processes and practical applications that underscore their significance in embedding tasks. By utilizing statistical techniques, data from high-dimensional spaces is condensed, preserving the inherent probabilistic connections and essential patterns as much as possible.

6.2.3.1 N-gram values

In reference to section 5.1.2, we adopt the use of n-gram values, specifically focusing on the frequency of byte combinations. However, an implication of this approach is that it leads to an exponentially high dimensional space. For instance, with a 2-gram, the potential values amount to $256 * 256 = 65536$. Given the extensive dimensionality, we have opted for combinations of bits rather than bytes. This change substantially reduces the space required; a 2-gram, in this case, would only amount to $2 * 2 = 4$ values.

Switching to bit combinations aligns well with our objectives. Our main interest is in the frequency patterns of n-gram values rather than the specific n-gram values themselves. This is because our core aim is to identify SSH keys, which inherently display frequencies for all combinations due to their random nature.

In our approach, we utilize 1-gram, 2-gram, 3-gram, and 8-gram values (of bits). The inclusion of 8-gram is particularly significant as it captures larger sequences, providing a broader context and enhancing the ability to discern patterns and anomalies that shorter n-grams might miss. Specifically, the 8-gram contributes 256 values to the embedding vector. When combined with the 8 values from the 3-gram, 4 from the 2-gram, and 2 from the 1-gram, the total dimensionality for the n-gram embedding becomes 270. This comprehensive approach ensures a more robust representation of the data, aiding in the accurate identification of SSH keys. Importantly, by opting for an 8-gram, we avoid the exponential growth in dimensionality that a larger n-gram, such as a 16-gram, would introduce, ensuring that the embedding vector remains manageable and doesn't explode in size or in memory usage.

6.2.3.2 Other statisticals values

In our approach, several metrics are employed to analyze the data. Specifically, we utilize the mean as detailed in 5.2, the standard deviation as found in 5.4, the MAD from 5.3, the skewness as outlined in 5.5, the kurtosis referenced in 5.6, and the Shannon entropy from 5.1. These metrics, when collectively considered, provide a comprehensive understanding and embed a plethora of information about the data at hand.

It's imperative to note a particular aspect of our analysis concerning the standard deviation. There are instances where the standard deviation registers a value of zero. Such an occurrence is indicative of

data consistency. Concurrently, in such scenarios, both the kurtosis and skewness are undefined. When faced with this situation, our course of action is to dismiss the chunk from our analysis. The rationale behind this is straightforward: a consistent chunk would likely not be pertinent to our exploration, especially when our aim is to identify patterns characteristic of an SSH key, wich are random by nature.

6.2.3.3 Statistical Embedding

For each chunk, we construct vectors using a combination of n-gram values and other statistical metrics. The n-gram approach contributes 270 distinct values to the vector. Simultaneously, the supplementary statistical metrics, which encapsulate measures of the mean, standard deviation, MAD, skewness, kurtosis, and Shannon entropy, introduce an additional 6 values. Furthermore, as discussed in a preceding section 6.2.2, the basic information associated with each chunk is also embedded into this vector. Consequently, the resultant vector for each structure comprises a total of 281 values, providing a comprehensive representation of the chunk's characteristics.

6.2.4 Graph embedding

In this section, we shift our focus towards the creation and embedding of graphs derived from the heap dump data. The process of graph creation involves structuring the data in a way that captures the relationships and connections between the chunks and their pointers. Subsequently, we will transform this graphs into low-dimensional vector representations, enabling the application of machine learning techniques to identifying chunks containing ssh keys.

6.2.4.1 Graphs creation

Our graph construction is a meticulously organized process aimed at representing the intricate relationships present within the heap dump data. Comprising three distinct node types - chunks, pointers, and value nodes - this graph provides a comprehensive view of the data's structure. Our approach commences with the sequential parsing of the heap dump data, enabling the identification of essential chunks central to our analytical objectives. These chunks form the core nodes of our graph. To establish connections between these chunks and the data they contain, we further divide each structure into 8-byte blocks, which is the size of the heap alignment. These blocks are then translated into value nodes within the graph, serving as connectors bridging the data structures to their specific data. An heuristic approach, grounded in REGEX 6.1.4.2, is employed to identify valid pointers within the heap dump data, with pointers representing a subset of value nodes, indicating legitimate pointers references. The scrupulously established connections between chunks, value nodes, and pointers ensure that the graph accurately mirrors the intricate relationships found within the heap dump data. This comprehensive graph construction process is efficiently implemented in Rust, making effective use of the Petgraph library to handle the complexities of heap dump data and graph representation, offering superior efficiency compared to a Python-based implementation.

In the following image 6.1, we can see the chunks nodes representing in blue, containing pointers nodes in orange and value nodes nodes in gray.

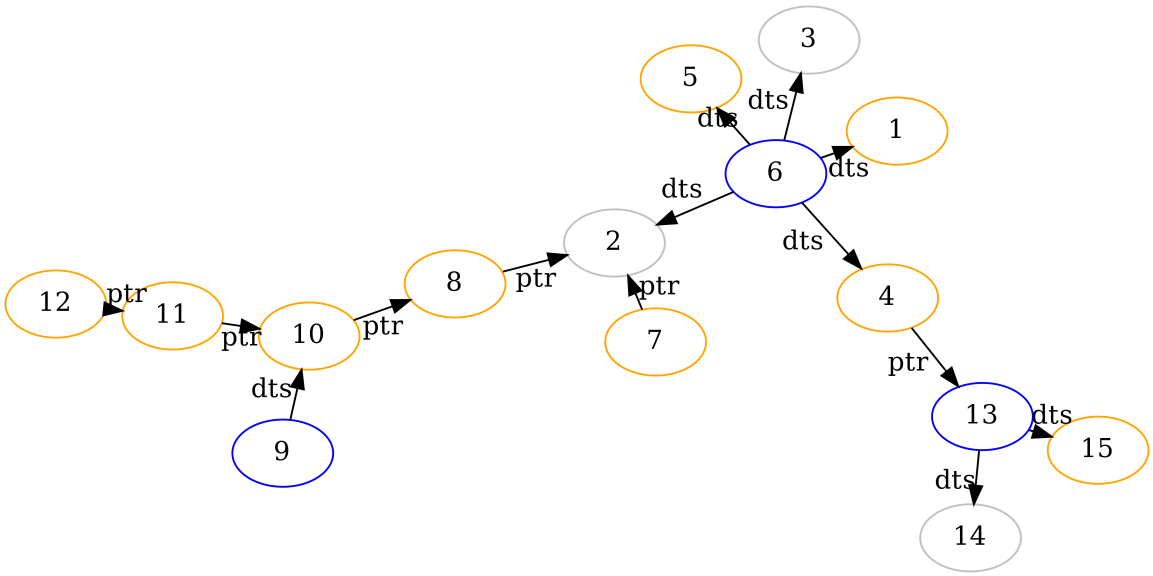


Figure 6.1: Graph creation process

After the construction of the graph, we can use graphviz (and the DOT language)[7] to visualize the graph, using the command :

```
sfdp -Gsize=67! -Goverlap=prism -Tpng dot_file > image.png
```

The following image is an example of the creation of the graph from the file `./Validation/Validation/basic/V_8_1_P1/24/27107-1643980590-heap.raw` without value nodes to enhance clarity.

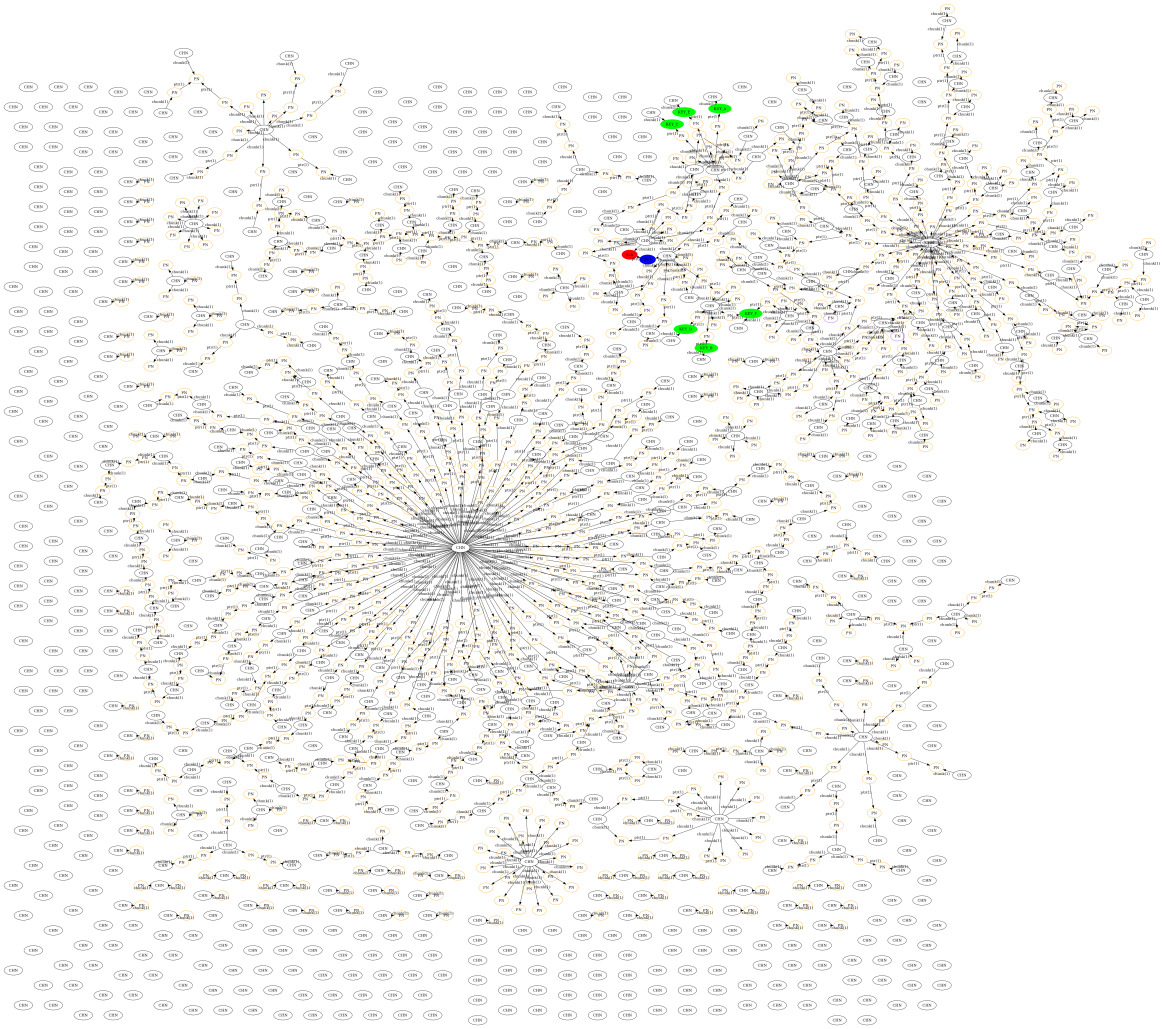


Figure 6.2: Graph example

6.2.4.2 Graphs embedding

Our next step is to uncover deeper insights and semantic understanding from our constructed graph, focusing on semantic embedding. This is the process through which we reshape our graph into a low-dimensional vector space, with each vector acting as a repository for a chunk's immediate neighborhood. Through this transformative journey, our aim is to forge vector representations that empower the application of cutting-edge machine learning techniques.

To create a concise yet informative representation, considering both structure-to-member and pointer-based connections, we meticulously count the number of pointers and chunks directly referencing a specific chunk's members. This initial count provides valuable insights into the chunk's immediate context. However, we doesn't stop there; we expand this representation by including counts of pointers and chunks pointing to those preceding nodes, allowing us to capture deeper layers of context. This recursive process continues until we reach a predetermined depth. Furthermore, we initiate a parallel analysis in reverse, meticulously tracing connections by following pointers from the initial chunk to capture its children, recursively delving deeper until we reach the specified depth. We can see the algorithm here 6.6. The result is a low-dimensional vector that intricately encodes the chunk's

neighborhood, offering a comprehensive view of its relationships and contextual significance within the graph.

Algorithm 6.6 Generate Ancestor/Children Embedding

```
function GenerateNeighborsCHN(chunk_node, dir)
    ancestor_nodes ← an empty set
    children ← graph.neighbors_directed(chunk_node, OUT)        ▷ Get members of the chunk
    for child in children do
        ancestor_nodes.insert(child)
    end for
    result ← an empty list
    current_nodes ← an empty set
    for _ in 0 to DEPTH do
        current_nodes ← ancestor_nodes                ▷ switch ancestor nodes and current nodes
        ancestor_nodes ← an empty set
        nb_chn ← 0
        nb_ptr ← 0
        for current_node in current_nodes do
            if node is ChunkHeaderNode then            ▷ Update number of chunks and pointers
                nb_chn ← nb_dtn + 1
            else if node is PointerNode then
                nb_ptr ← nb_ptr + 1
            end if                                     ▷ Get neighbors of the current node
            for neighbor in graph.neighbors_directed(current_node, dir) do
                ancestor_nodes.insert(neighbor)        ▷ Add neighbors to the next ancestor nodes
            end for
        end for
        result.append(nb_chn)                          ▷ Add number of data structures
        result.append(nb_ptr)                          ▷ Add number of pointers
    end for
    return result
end function
```

We can apply this algorithm to every chunk within each graph, delving to a depth of 8, which produces an embedding of 32 units: 8 for ancestor pointers, 8 for ancestor chunks, 8 for child pointers, and 8 for child chunks. To accurately represent the chunk's neighborhood, it's crucial not to omit details about its members. Thus, we incorporate the basic chunk information, which includes the block position in the chunk, chunk byte size, number of pointers in the chunk, number of value nodes in the chunk, and the chunk's index in the heap. This results in a final embedding size that is an aggregate of the neighborhood representation and the basic chunk information, summing up to 37 values. However, there are inherent challenges with this embedding. It tends to get polluted by the value node, which often lacks significant meaning. Moreover, the relationships between the structures are intricate, and there's potential to represent them in a more straightforward manner, as shown in the next section.

6.2.4.3 Updated graph

Recognizing these challenges and the need for a clearer representation, we embarked on a series of refinements. Our approach focuses on enhancing the last graph by preserving the structure nodes and their interconnections via pointers. To simplify the visualization, we've decided to eliminate both the value nodes and the pointer nodes. In addition, the relationships that previously connected the pointer nodes to the value nodes will now link directly to the chunk nodes, with the added detail of weighted edges. This strategy is driven by our aspiration to offer a more lucid graph, significantly reducing any extraneous noise, as show in the figure 6.3, the representation of the same file `./Validation/Validation/basic/V_8_1_P1/24/27107-1643980590-heap.raw`.

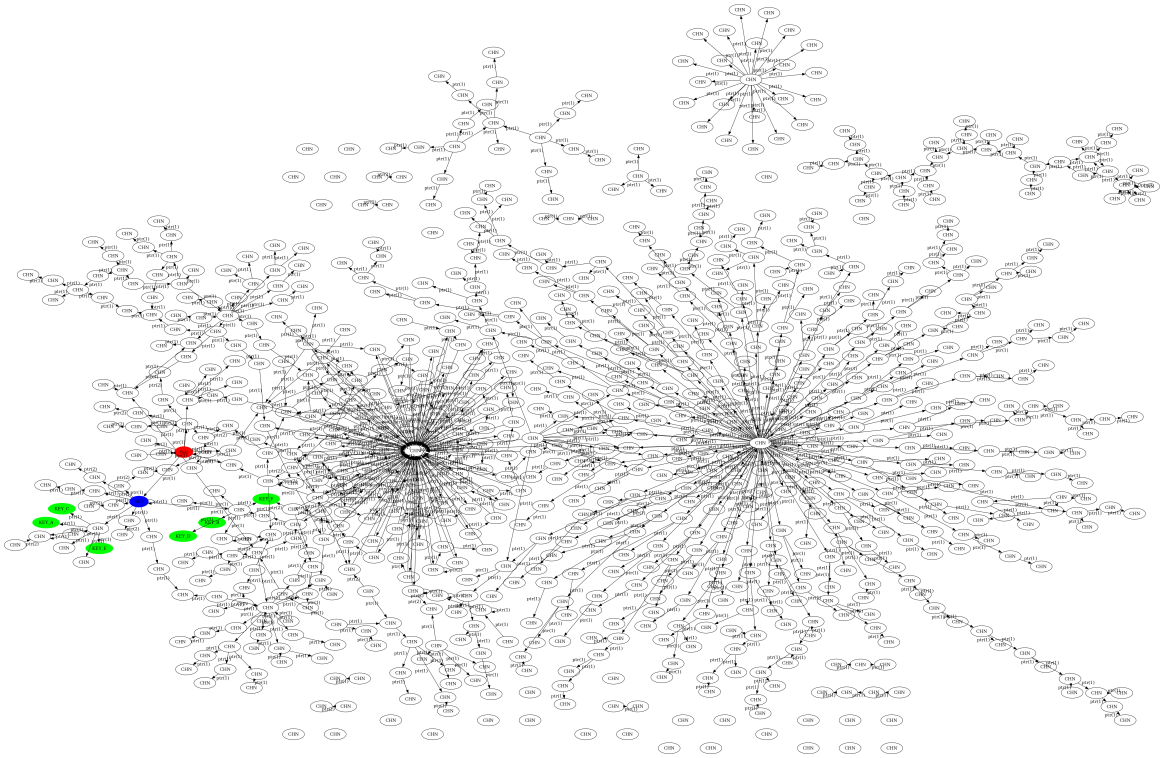


Figure 6.3: Updated graph

By eliminating the pointer nodes, we have successfully integrated their information directly into the relationships between chunk nodes. The embedding now includes 8 preceding depth chunk nodes, 8 subsequent depth chunk nodes, and 5 basic information values, as outlined in section 6.2.2. This consolidation results in a compact representation of 21 values, enhancing the embedding's simplicity and efficiency. Although the graph now communicates information more effectively, the embedding has become less detailed. This change in richness could potentially impact the performance of machine learning tasks.

6.2.4.4 Other Graph Embedding Techniques

In this work, our focus is primarily on the embedding techniques previously discussed, and we will not delve into more advanced graph embedding methods such as deep learning-based node2vec or Graph

Convolutional Networks (GCN). These sophisticated techniques offer a different approach to graph embedding, capturing complex patterns and relationships within the data. For readers interested in a more comprehensive exploration of these advanced methods, especially in the context of detecting SSH keys, the master thesis of Florian Rascoussier provides an in-depth analysis and application of these techniques.

6.2.5 Raw User Data Embedding

In this section, we delve into the embedding of raw user data for each chunk, with the specific aim of identifying chunks that contain SSH keys. This task presents unique challenges, primarily due to the variability in data size across different chunks. Some chunks can be significantly large, with sizes reaching up to 60,000 bytes, which adds a layer of complexity to the embedding process.

To tackle the challenges associated with embedding raw user data for chunk analysis, we integrate a variety of techniques into a cohesive approach. Simple extraction serves as a foundational method, standardizing data size through padding and trimming to create uniform embeddings, which simplifies subsequent analysis. Complementing this, we incorporate advanced embedding techniques like Word2Vec and transformers. Word2Vec, borrowed from natural language processing, helps decipher patterns and relationships within raw byte sequences. Transformers, renowned for their ability to capture long-range dependencies and intricate patterns, are particularly adept at handling the complexity of large data chunks, some of which can be as extensive as 60,000 bytes. Together, these methods form a robust framework, ensuring that we can effectively identify and isolate chunks containing SSH keys, regardless of their size or the complexity of the embedded data.

6.2.5.1 Simple Extraction

In the realm of raw user data chunk embedding, two straightforward techniques stand out: trimming and padding. These methods provide a basic yet effective means of standardizing the size of the data for embedding, which is crucial for consistent analysis.

Trim Method:

The trim method involves cutting down the raw user data to a predetermined fixed size. Given that SSH keys are always located at the beginning of a chunk, and considering the minimum size of a key is 12 bytes and the minimum chunk size is 16 bytes, we can confidently trim all user data to 12 bytes. This results in 1 byte per feature, and when combined with the 5 basic chunk information features, the total comes to 17 features. This method ensures a uniform data size, facilitating a more streamlined and efficient embedding process.

Padding Method:

The padding method, on the other hand, involves augmenting smaller data sequences with additional zeros to match the size of the larger ones. While this technique ensures uniformity in data size, it results in significantly larger embeddings. These large embeddings can be cumbersome and challenging to work with, making them less practical for our purposes. As a result, we must explore alternative techniques to achieve our goal of efficient and useful data representation.

6.2.5.2 Word2Vec

Word2Vec, extensively discussed in section 5.2, stands out as a flexible tool originally crafted for identifying patterns within text data. Its capabilities, however, transcend textual analysis, rendering it apt for sequences of raw user data. In such scenarios, we interpret sequences of hexadecimal characters as words, while chunks are analogous to sentences.

To augment the Word2Vec model's effectiveness, we can apply previously delineated filters for entropy and chunk size. The primary goal here is to leverage Word2Vec to generate comprehensive embedding vectors for each chunk, effectively converting them into meaningful sentences within the embedding space.

Word2Vec assigns a feature vector to each word. To derive the embedding for an entire sentence—or a chunk in our context—we calculate the mean of the feature vectors for all words in the sentence. Algorithm 6.7 elucidates this process, illustrating the transition from individual word embeddings to a collective sentence embedding. It is crucial to acknowledge that the model disregards out-of-vocabulary words, namely, words present in the dataset to be embedded but not encountered during the Word2Vec model's training phase.

Algorithm 6.7 Get Average Word Embedding (word2vec)

```
1: function GetAverageEmbedding(word_sequence, model)   ▷ Model are the features for each word
2:     embeddings ← empty list
3:     for each word in word_sequence do
4:         if word in model.wv then
5:             embeddings.append(model.wv[word])
6:         end if
7:     end for
8:     if length of embeddings > 0 then
9:         return mean(embeddings)                       ▷ mean for each feature
10:    else
11:        return vector of size model.vector_size of zeros
12:    end if
13: end function
```

In our research, we employ distinct datasets for training, validation, and testing. The Word2Vec model is trained using the training dataset, and this trained model is then used to embed the validation and testing datasets. This approach, implemented using the Python package gensim [29] [1], may not

[1] `https://pypi.org/project/gensim/`

guarantee optimal performance due to the concurrent use of the same dataset for both training and embedding. However, it serves our purpose of comparing and analyzing various embedding techniques. Several hyperparameters are employed in this process, such as the embedding dimension, which will be further reduced to 8 to match other embeddings, the window size of the algorithm, and the word size. These hyperparameters are detailed in the annex. To minimize computation time and memory usage, we opt for lower values of hyperparameters, although a range of values are tested. It is worth noting that there is potential for further refinement of these parameters. For instance, considering a block is 8 bytes, one could set the window size to 8 bytes, or adjust it to the size of the smallest key, which is 12 bytes. However, such refinements are beyond the scope of this thesis.

6.2.5.3 Transformers and Head Attention Algorithm

As we transition from the Word2Vec model, we explore the transformers and head attention algorithm, another influential model in the realm of natural language processing (NLP). In this model, sequences of hexadecimal characters are interpreted as words, while chunks are viewed as sentences. A distinctive feature of transformers is the requirement for all sentences to be of equal length, necessitating the padding of shorter sentences.

Implemented in Python using the Keras library [2], the architecture and parameters of the neural network are detailed in the annexes. The output of the neural network serves as the embedding, which can optionally be followed by a classifier neural network. For consistency in evaluation, we utilize the generated embedding in the same manner as other embeddings. In this implementation, the training and embedding processes occur simultaneously within the transformers model, ensuring consistent processing of both the training and validation datasets. This approach is chosen because we are using the neural network primarily as an embedding technique rather than a classifier. However, it is possible to add an additional layer to the neural network to transform it into a classifier model, compile it, and then train/fit it, potentially accelerating the training process and allowing it to function similarly to Word2Vec or a random forest model.

Similar to the Word2Vec model, the performance of the transformers model is contingent on the selection of hyperparameters. To mitigate computation time and memory usage, we opt for lower values of hyperparameters, while ensuring a comprehensive evaluation through testing a range of values. The key hyperparameters in our transformers implementation include:

- **Transformer Units:** Number of units or neurons in the dense layers inside the transformer encoder, determining the dimensionality of the outputs of the multi-head attention mechanism and the size of the feed-forward neural network layers.
- **Number of Heads:** Specifies the number of attention heads, influencing the model's capacity to focus on different segments of the input sequence.

[2] `https://keras.io/`

- **Dropout Rate:** Defines the dropout rate applied to the attention scores, aiding in the prevention of overfitting.
- **Number of Transformer Layers:** Determines the number of transformer layers stacked in the model, each layer enhancing the model's ability to capture complex data patterns.
- **Activation Function:** Specifies the activation function used in the dense layers inside the transformer encoder, introducing non-linearity to the model.

A notable distinction in our transformers implementation is the use of float64 as the input data type, which inherently limits the word size to 64 bits (8 bytes) for conversion into float64. This choice significantly increases the amount of memory required to run the model, subsequently constraining the size of the dataset that can be processed. This trade-off highlights the challenges and considerations involved in implementing complex NLP models like transformers, especially when dealing with large volumes of data.

6.3 Embedding quality

The quality of embeddings is paramount in machine learning, particularly when the objective is to identify specific chunks within data, such as the ones holding SSH keys. It becomes essential to juxtapose the performances of all embeddings in this context. An optimal embedding should proficiently discern the chunks containing SSH keys across the entire spectrum of openSSH use cases and for every conceivable key size. This necessitates the utilization of the complete dataset, with the training subset dedicated to model training and the validation subset for testing. Addressing this from a machine learning classification perspective, the random forest model, as elucidated in 5.3.4, emerges as the classifier of choice.

To ensure fairness and comparability among the embeddings, we employ the Pearson correlation method 5.3.2.1 to limit the selection to the top 8 correlations, thereby narrowing down our analysis to the most influential features. The dataset is notably imbalanced 5.3.3, primarily stemming from the rarity of memory structures containing SSH keys, our specific target of interest, within the overall dataset. This rarity results in a significant class imbalance, where the majority of memory structures do not contain SSH keys. To counteract potential bias toward the majority class, we will implement random undersampling as a resampling strategy, particularly given our very large dataset. This approach will enable our model to accurately classify both majority and minority classes without being overwhelmed by the sheer volume of data. We will then employ a Random Forest model 5.3, renowned for its robustness and suitability for high-dimensional data, to carry out the classification task. Our evaluation will rely on metrics such as precision, recall, F1 score, and others to identify the most effective representation for precise classification.

6.3.1 Feature Selection and Dataset Challenges

In the quest for fairness across various embeddings and to circumvent the curse of dimensionality, it's imperative to maintain a uniform feature count across all embeddings. This is where feature engineering shines. The Pearson correlation method, elaborated in 5.3.2.1, is harnessed to meticulously select the 8 most salient features for each embedding. This count is a judicious compromise, ensuring the features are both succinct in number and information-rich. However, the dataset presents its own set of challenges. The instances of chunks containing SSH keys are dwarfed by those devoid of them, leading to a pronounced dataset imbalance. To counteract this skewness, the random undersampling technique, as referenced in 5.3.3, is employed, when the dataset isn't filtered.

6.3.2 Implementation and Evaluation Metrics

The implementation leans heavily on the scikit-learn library [26] in Python, which provides the tools for the random forest classifier, Pearson correlation, and the random undersampling algorithm. Concurrently, the pandas library is indispensable for the efficient loading and manipulation of the dataset. Before diving into the analysis, it's crucial to ensure the embedding's integrity. This involves a rigorous sanity check, especially given the potential for corruption, such as NaN values. To guarantee

the reproducibility of results, a consistent random seed is employed for both the random forest classifier and the random undersampling algorithm. For a comprehensive evaluation, the Pearson correlation matrix is preserved for each embedding. Moreover, a suite of metrics, including precision, recall, f1-score, AUC, and the confusion matrix (encompassing true positives, true negatives, false positives, and false negatives), is meticulously saved for every embedding.

6.4 Embedding Coherence

A significant facet of this thesis revolves around the comparison of embeddings derived from various use cases, versions, and key sizes of OpenSSH. The objective is to discern whether there exists a coherent relationship among them. To facilitate this comparison, a clustering algorithm is employed to categorize the different embeddings and assess their mutual coherence.

6.4.1 Detailed Clustering Approach

For the clustering of embeddings, we opt for the OPTICS algorithm, as referenced in section 5.4.4. OPTICS is particularly well-suited for this task due to its proficiency in handling data with varying cluster densities, a common characteristic of our embedding data.

In our clustering approach, we utilize the cosine distance metric. This choice is strategic, as it mitigates the challenges posed by the curse of dimensionality, eliminating also the need for data scaling prior to clustering. The cosine distance provides a measure of similarity that is not influenced by the magnitude of the data, focusing solely on the direction of the data points in the high-dimensional space.

To determine the optimal number of clusters for each embedding, we employ the xi parameter of the OPTICS algorithm. A range of xi values is tested, and the configuration yielding the highest silhouette score is selected. The silhouette score serves as a quantitative measure of the quality of the clustering, with higher values indicating more distinct and well-separated clusters.

In our implementation, we resort to the brute force method to compute the cosine metrics, ensuring accuracy in our distance calculations at the expense of computational efficiency. This method systematically calculates the cosine distance between all possible pairs of data points, providing a comprehensive assessment of the similarities and differences among the embeddings

6.4.2 Limits and Adaptation

Clustering algorithms, while powerful, come with their own set of challenges, particularly in terms of computational demands. They are known for their high memory consumption and intensive calculations, which can pose constraints when dealing with large datasets. As a result, there's often a need to limit the number of input data points or samples to ensure efficient processing.

A straightforward, albeit non-optimal, strategy to reduce the number of samples is random sampling. While this approach might not capture the full diversity and nuances of the dataset, it offers a feasible starting point for preliminary analyses. To maintain coherence and representativeness in the sampled

data, it's essential to preserve the ratio of significant labels, such as keys and SSH structures (SSH_-STRUCT and SESSION_STATE), to noise points. This ensures that the key characteristics of the dataset are retained in the sample.

However, it's worth noting that this sampling method doesn't guarantee that all possible file variations are represented in the sample. Despite its limitations, random sampling serves as an initial approach, providing a snapshot of the dataset's characteristics and offering insights that can guide further, more detailed analyses.

7 Results

Describe the experimental setup, the used datasets/parameters and the experimental results achieved

8 Discussion

8.1 Limits

This study, while comprehensive, acknowledges several limitations that warrant mention. Within the realm of data embedding, a notable constraint is the necessity to deactivate the entropy filter during the validation phase to prevent the inadvertent exclusion of keys. Additionally, the NLP models, such as Word2Vec and transformers, are highly sensitive to hyperparameter tuning. Due to time constraints, we have explored only a limited subset of these parameters, which may impact the robustness of our findings.

Memory consumption poses another significant challenge, particularly for NLP models and some simpler embedding techniques. The computational resources available for this study, specifically the server with 516 GB of RAM, restricted our ability to process some of the more demanding models.

In the evaluation of embedding performance, the decision to limit features to eight was arbitrary and could have been adjusted to better suit the characteristics of each embedding. While the Pearson algorithm was chosen for its ease of implementation, more sophisticated dimensionality reduction techniques, such as PCA, might have yielded more nuanced insights. Furthermore, the exclusion of time as a factor in the analysis was a deliberate choice to maintain consistency across models. However, time is a critical element in practical applications, such as SSH key detection, and should be considered in future evaluations.

The section on embedding coherence also presents its own set of challenges. Clustering, while a powerful tool, is difficult to interpret and optimize, requiring careful tuning of numerous hyperparameters. The time and memory demands of clustering have limited the depth of exploration in this thesis. Consequently, the results obtained were not as robust as desired, and the methods employed to manage data volume, such as random undersampling, were not ideal.

8.2 Future Work

Looking ahead, there is ample opportunity for further research and enhancement in several areas. The embedding techniques could benefit from the application of more sophisticated or varied NLP models, along with extensive hyperparameter tuning. Improvements in the performance evaluation of embeddings could be achieved through the use of more advanced dimensionality reduction algorithms and a more detailed examination of processing time to better differentiate between models.

The exploration of embedding coherence in this thesis is merely an initial foray. Significant improvements could be realized with a more refined approach to dataset preparation and further tuning of

clustering algorithms. As such, these areas present fruitful avenues for future research and development.

9 Conclusion

Summarize the thesis and provide a outlook on future work.

10 Ressources

TODO : make transition

10.1 hardware

My primary workstation is an *Aspire 5* laptop, equipped with:

- **CPU:** 11th Gen Intel i5-1135G7 (8) @ 4.200GHz
- **GPU:** Intel TigerLake-LP GT2 [Iris Xe Graphics]
- **Memory:** 16GB

However, this laptop, despite its decent specifications, proved inadequate for processing the entire dataset. Simple machine learning experiments using a Python script would have stretched over a week. Even when we transitioned to more optimized Rust programs, the processing time exceeded 10 hours. While I managed to run minor tasks and scripts on this laptop, the bulk of the experiments necessitated a more powerful server.

Recognizing this need, I was granted access to a high-performance development server in the later stages of the thesis, around August 2023. The server, an *AS-4124GS-TNR*, boasts the following specifications:

- **CPU:** 2x AMD EPYC 7662 (256) @ 2.000GHz
- **GPU:** NVIDIA Geforce RTX 3090 Ti
- **RAM:** 512GB DDR4 3200MHz

Operating on *Ubuntu 20.04.6 LTS*, this server became the primary platform for the machine learning experiments, given its superior computational capabilities compared to the *Aspire 5* laptop. This invaluable resource was generously provided by the Department of Computer Science at *Universität Passau*, particularly under the guidance of the Chair of Data Science led by Prof. Dr. Michael Granitzer. I extend my sincere appreciation for their unwavering support.

A Models

A.1 Machin Learning Hyperparameters

A.1.1 Random Forest Classifier

Table A.1: Default Parameters for Random Forest Classifier

Parameter	Default Value
bootstrap	True
ccp_alpha	0.0
class_weight	None
criterion	gini
max_depth	None
max_features	sqrt
max_leaf_nodes	None
max_samples	None
min_impurity_decrease	0.0
min_samples_leaf	1
min_samples_split	2
min_weight_fraction_leaf	0.0
n_estimators	100
n_jobs	-1
oob_score	False
random_state	42
verbose	0
warm_start	False

A.1.2 OPTICS Clustering

Table A.2: Default Parameters for OPTICS Clustering

Parameter	Default Value
algorithm	brute
cluster_method	xi
leaf_size	30
max_eps	inf
memory	None
metric	cosine
metric_params	None
min_cluster_size	None
n_jobs	-1
p	2
predecessor_correction	True
xi	0.05

Note for OPTICS:

min_samples: Calculated dynamically for each embedding.

eps: Takes five distinct values: 0.01, 0.02, 0.03, 0.04, and 0.05.

A.2 Deep learning hyperparameters

A.2.1 Transformers:

Table A.3: Transformers Hyperparameters (Configurations 0–4)

	Config 0	Config 1	Config 2	Config 3	Config 4
word character size	16	16	8	8	16
embedding dim	8	16	8	16	8
transformer units	2	2	2	2	4
num heads	2	2	2	2	4
num transformer layers	2	2	2	2	4
dropout rate	0.1	0.1	0.1	0.1	0.3
activation	relu	relu	relu	relu	relu

Table A.4: Transformers Hyperparameters (Configurations 5–7)

	Config 5	Config 6	Config 7
word character size	16	8	8
embedding dim	16	8	16
transformer units	4	4	4
num heads	4	4	4
num transformer layers	4	4	4
dropout rate	0.3	0.3	0.3
activation	relu	relu	relu

A.2.2 Word2Vec:

Table A.5: Word2Vec Hyperparameters (Configurations 0–4)

	Config 0	Config 1	Config 2	Config 3	Config 4
output size	8	8	8	8	16
window character size	8	8	16	16	8
word character size	2	4	2	4	2
min count	1	1	1	1	1

B Dataset

B.1 Dataset cleaning results

The empty folder for the training part of the dataset after cleaning are :

Table A.6: Word2Vec Hyperparameters (Configurations 5–9)

	Config 5	Config 6	Config 7	Config 8	Config 9
output size	16	16	16	100	100
window character size	8	16	16	8	8
word character size	4	2	4	2	4
min count	1	1	1	1	1

Table A.7: Word2Vec Hyperparameters (Configurations 10–11)

	Config 10	Config 11
output size	100	100
window character size	16	16
word character size	2	4
min count	1	1

Table B.1: List of empty Folders in the training subdataset Categorized by OpenSSH Parameters

Use Case	**Version**	**Key Size**
port-forwarding	V_8_0_P1	64
port-forwarding	V_8_0_P1	32
port-forwarding	V_7_8_P1	16
port-forwarding	V_7_8_P1	64
port-forwarding	V_7_8_P1	32
scp	V_8_0_P1	64
scp	V_8_0_P1	32
scp	V_7_8_P1	64
scp	V_7_8_P1	32
basic	V_8_7_P1	16
basic	V_8_7_P1	64
basic	V_8_7_P1	32
basic	V_8_8_P1	16
basic	V_8_8_P1	64
basic	V_8_8_P1	32
basic	V_7_0_P1	16
basic	V_7_0_P1	64
basic	V_7_0_P1	32
basic	V_6_8_P1	16
basic	V_6_8_P1	64
basic	V_6_8_P1	32
basic	V_6_2_P1	16
basic	V_6_2_P1	24
basic	V_6_2_P1	32
basic	V_6_0_P1	16
basic	V_6_0_P1	24

Continued on next page

Table B.1 – continued from previous page

Use Case	Version	Key Size
basic	V_6_0_P1	32
basic	V_8_1_P1	16
basic	V_8_1_P1	64
basic	V_8_1_P1	32
basic	V_6_1_P1	16
basic	V_6_1_P1	24
basic	V_6_1_P1	32
basic	V_7_2_P1	16
basic	V_7_2_P1	64
basic	V_7_2_P1	32
basic	V_8_0_P1	16
basic	V_8_0_P1	64
basic	V_8_0_P1	32
basic	V_6_3_P1	16
basic	V_6_3_P1	24
basic	V_6_3_P1	32
basic	V_6_9_P1	16
basic	V_6_9_P1	64
basic	V_6_9_P1	32
basic	V_7_1_P1	16
basic	V_7_1_P1	64
basic	V_7_1_P1	32
basic	V_7_9_P1	16
basic	V_7_9_P1	64
basic	V_7_9_P1	32
basic	V_6_7_P1	16
basic	V_6_7_P1	24
basic	V_6_7_P1	32
basic	V_7_8_P1	16
basic	V_7_8_P1	64
basic	V_7_8_P1	32
client	V_8_0_P1	16
client	V_8_0_P1	64
client	V_8_0_P1	32
client	V_7_8_P1	16
client	V_7_8_P1	64
client	V_7_8_P1	32

The empty folder for the validation part of the dataset after cleaning are :

Table B.2: List of empty Folders in the validation subdataset Categorized by OpenSSH Parameters

Use Case	Version	Key Size
port-forwarding	V_8_0_P1	64
port-forwarding	V_8_0_P1	32
port-forwarding	V_7_8_P1	16
port-forwarding	V_7_8_P1	64
port-forwarding	V_7_8_P1	32
scp	V_8_0_P1	64
scp	V_8_0_P1	32
scp	V_7_8_P1	64
scp	V_7_8_P1	32
basic	V_8_7_P1	16
basic	V_8_7_P1	64
basic	V_8_7_P1	32
basic	V_8_8_P1	16
basic	V_8_8_P1	64
basic	V_8_8_P1	32
basic	V_7_0_P1	16
basic	V_7_0_P1	64
basic	V_7_0_P1	32
basic	V_6_8_P1	16
basic	V_6_8_P1	64
basic	V_6_8_P1	32
basic	V_6_2_P1	16
basic	V_6_2_P1	24
basic	V_6_2_P1	32
basic	V_6_0_P1	16
basic	V_6_0_P1	24
basic	V_6_0_P1	32
basic	V_8_1_P1	16
basic	V_8_1_P1	64
basic	V_8_1_P1	32
basic	V_6_1_P1	16
basic	V_6_1_P1	24
basic	V_6_1_P1	32
basic	V_7_2_P1	16
basic	V_7_2_P1	64
basic	V_7_2_P1	32
basic	V_8_0_P1	16
basic	V_8_0_P1	64
basic	V_8_0_P1	32

Continued on next page

Table B.2 – continued from previous page

Use Case	Version	Key Size
basic	V_6_3_P1	16
basic	V_6_3_P1	24
basic	V_6_3_P1	32
basic	V_6_9_P1	16
basic	V_6_9_P1	64
basic	V_6_9_P1	32
basic	V_7_1_P1	16
basic	V_7_1_P1	64
basic	V_7_1_P1	32
basic	V_7_9_P1	16
basic	V_7_9_P1	64
basic	V_7_9_P1	32
basic	V_6_7_P1	16
basic	V_6_7_P1	24
basic	V_6_7_P1	32
basic	V_7_8_P1	16
basic	V_7_8_P1	64
basic	V_7_8_P1	32
client	V_8_0_P1	16
client	V_8_0_P1	64
client	V_8_0_P1	32
client	V_7_8_P1	16
client	V_7_8_P1	32

The folders left for the training part of the dataset after cleaning are :

Table B.3: List of kept Folders in the Training subdataset Categorized by OpenSSH Parameters

Use Case	Version	Key Size
port-forwarding	V_8_0_P1	16
port-forwarding	V_8_0_P1	24
port-forwarding	V_7_8_P1	24
scp	V_8_0_P1	16
scp	V_8_0_P1	24
scp	V_7_8_P1	16
scp	V_7_8_P1	24
basic	V_8_0_P1	24
basic	V_7_8_P1	24
basic	V_7_1_P1	24
basic	V_7_0_P1	24

Continued on next page

Table B.3 – continued from previous page

Use Case	Version	Key Size
basic	V_7_9_P1	24
basic	V_8_1_P1	24
basic	V_6_9_P1	24
basic	V_8_7_P1	24
basic	V_8_8_P1	24
basic	V_6_8_P1	24
basic	V_7_2_P1	24
client	V_8_0_P1	24
client	V_7_8_P1	24

The folders left for the validation part of the dataset after cleaning are :

Table B.4: List of kept Folders in the Validation subdataset Categorized by OpenSSH Parameters

Use Case	Version	Key Size
port-forwarding	V_8_0_P1	16
port-forwarding	V_8_0_P1	16
port-forwarding	V_8_0_P1	24
port-forwarding	V_7_8_P1	24
scp	V_8_0_P1	16
scp	V_8_0_P1	24
scp	V_7_8_P1	16
scp	V_7_8_P1	24
basic	V_8_0_P1	24
basic	V_7_8_P1	24
basic	V_7_1_P1	24
basic	V_7_0_P1	24
basic	V_7_9_P1	24
basic	V_8_1_P1	24
basic	V_6_9_P1	24
basic	V_8_7_P1	24
basic	V_8_8_P1	24
basic	V_6_8_P1	24
basic	V_7_2_P1	24
client	V_8_0_P1	24
client	V_7_8_P1	24

The folders left for the performance test part of the dataset after cleaning are :

Table B.5: List of kept Folders in the Performance Test sub-dataset Categorized by OpenSSH Parameters

Version	Key Size
V_8_0_P1	32
V_8_0_P1	16
V_8_0_P1	24
V_7_8_P1	32
V_7_8_P1	16
V_7_8_P1	24
V_7_1_P1	32
V_7_1_P1	16
V_7_1_P1	24
V_7_9_P1	32
V_7_9_P1	16
V_7_9_P1	24
V_8_1_P1	32
V_8_1_P1	16
V_8_1_P1	24

C Machin Learning Results

C.1 Timeout instances

dataset	instance
25_filtered_chunk_extraction_-e_none_-s_activate	Transformers 0
25_filtered_chunk_extraction_-e_none_-s_activate	Transformers 1
26_filtered_chunk_extraction_-e_only-max-entropy_-s_none	Transformers 2
26_filtered_chunk_extraction_-e_only-max-entropy_-s_none	Transformers 3
26_filtered_chunk_extraction_-e_only-max-entropy_-s_none	Transformers 4
26_filtered_chunk_extraction_-e_only-max-entropy_-s_none	Transformers 5
26_filtered_chunk_extraction_-e_only-max-entropy_-s_none	Transformers 6
26_filtered_chunk_extraction_-e_only-max-entropy_-s_none	Transformers 7

Table C.1: Timeouts instances

C.2 Feature engineering fails

The list is empty.

C.3 Feature Engineering results

C.3.1 25_filtered_chunk_extraction_-e_none_-s_activate

Table C.2: Word2vec 0 Feature Engineering Results on 25_-filtered_chunk_extraction_-e_none_-s_activate

Dataset Name	25_filtered_chunk_extraction_-e_none_-s_activate
Instance	Word2vec 0
Best Features	feature_3
	feature_6
	feature_7
	feature_5
	feature_1
	feature_4
	feature_2
	feature_0

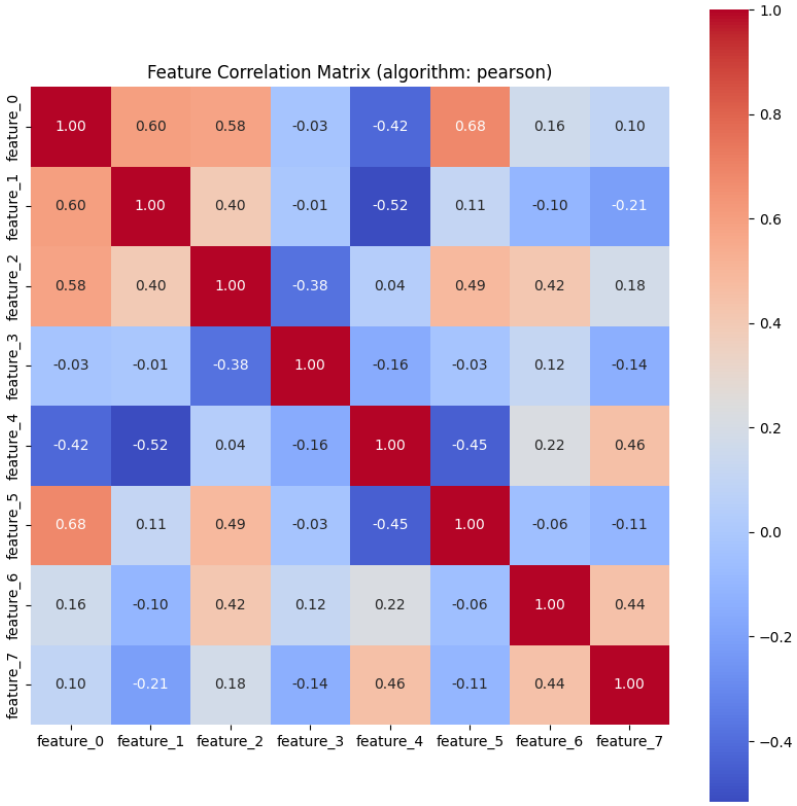


Table C.3: Word2vec 1 Feature Engineering Results on 25_-filtered_chunk_extraction_-e_none_-s_activate

Dataset Name	25_filtered_chunk_extraction_-e_none_-s_activate
Instance	Word2vec 1
Best Features	feature_7
	feature_3
	feature_4
	feature_0
	feature_5
	feature_6
	feature_2
	feature_1

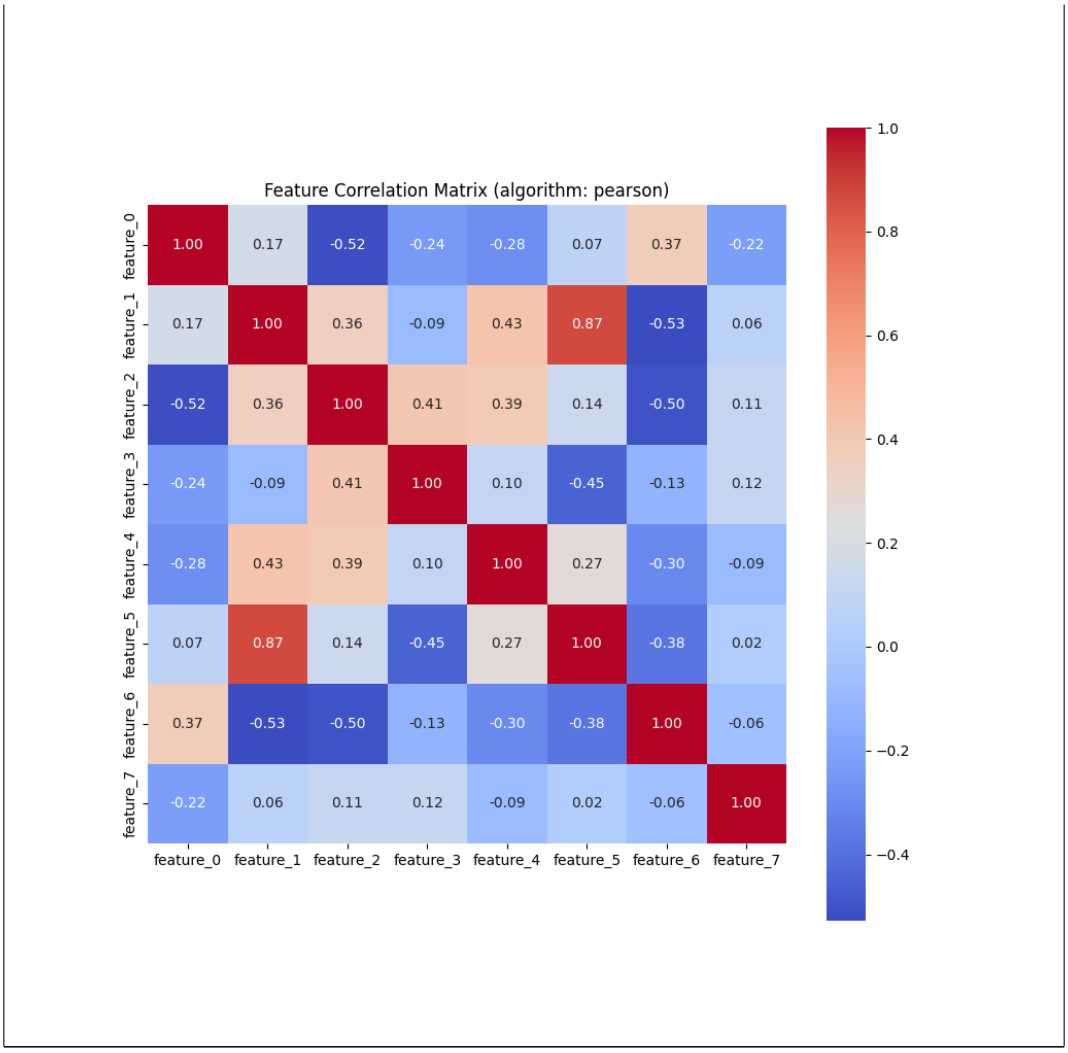


Table C.4: Word2vec 2 Feature Engineering Results on 25_-filtered_chunk_extraction_-e_none_-s_activate

Dataset Name	25_filtered_chunk_extraction_-e_none_-s_activate
Instance	Word2vec 2

Best Features	feature_3
	feature_7
	feature_5
	feature_4
	feature_1
	feature_6
	feature_2
	feature_0

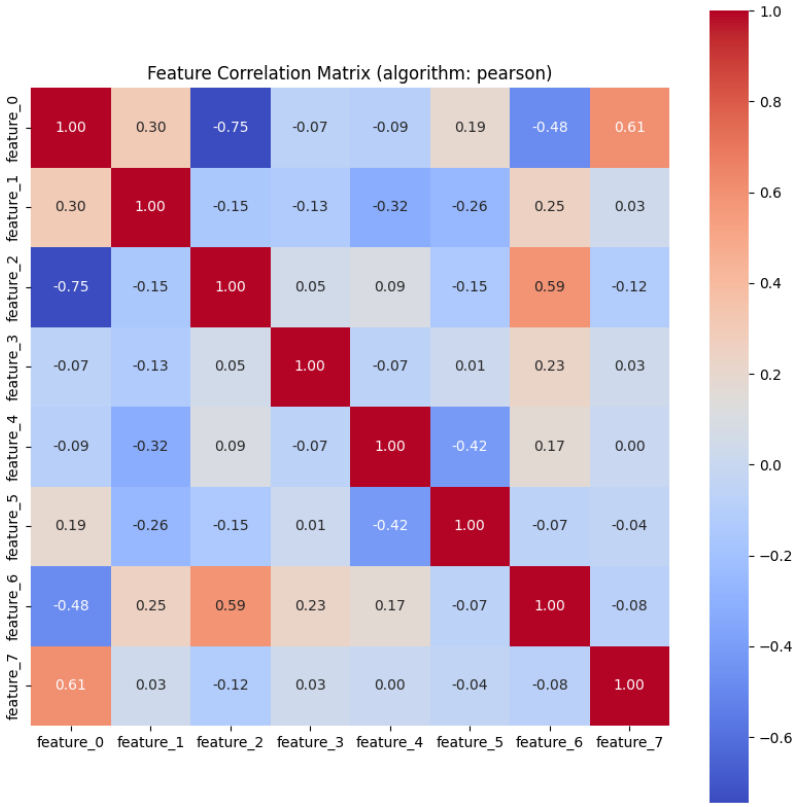


Table C.5: Word2vec 3 Feature Engineering Results on 25_-filtered_chunk_extraction_-e_none_-s_activate

Dataset Name	25_filtered_chunk_extraction_-e_none_-s_activate
Instance	Word2vec 3
Best Features	feature_2
	feature_5
	feature_3
	feature_6

feature_7
feature_4
feature_0
feature_1

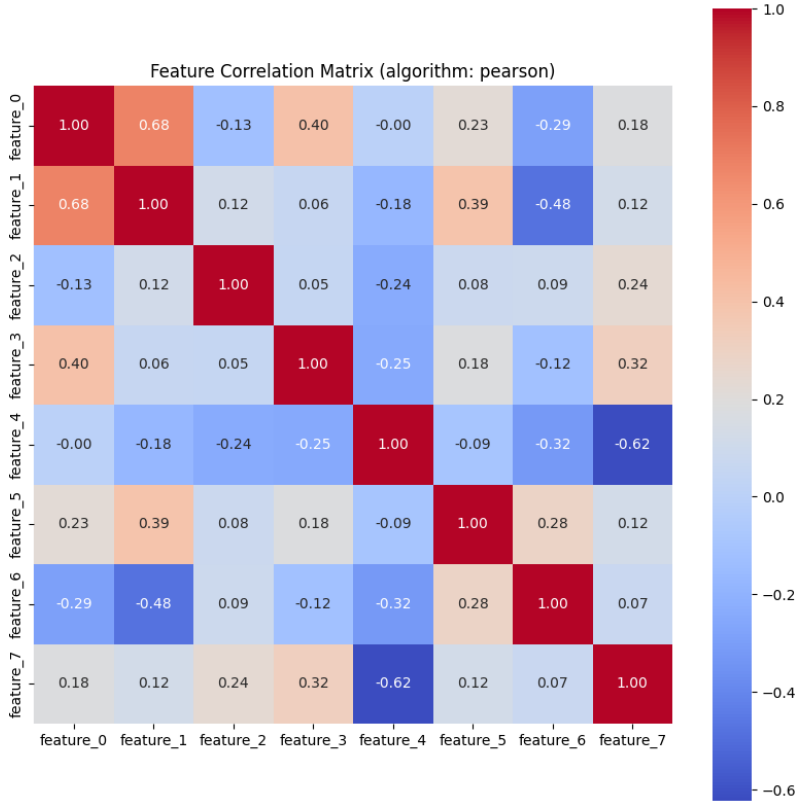


Table C.6: Word2vec 4 Feature Engineering Results on 25_-filtered_chunk_extraction_-e_none_-s_activate

Dataset Name	25_filtered_chunk_extraction_-e_none_-s_activate
Instance	Word2vec 4
Best Features	feature_1
	feature_4
	feature_10
	feature_8
	feature_13
	feature_15
	feature_0
	feature_5

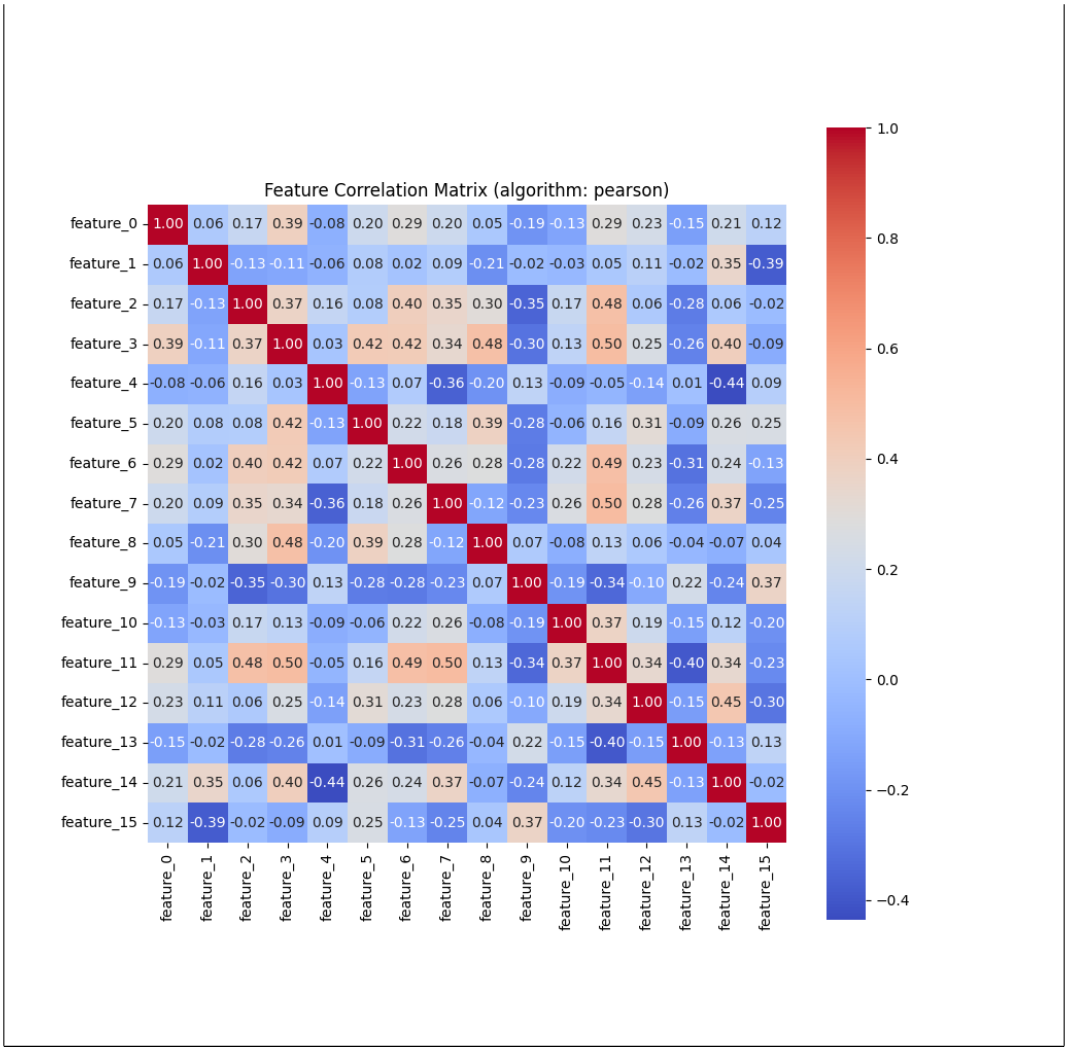


Table C.7: Word2vec 5 Feature Engineering Results on 25_-filtered_chunk_extraction_-e_none_-s_activate

Dataset Name	25_filtered_chunk_extraction_-e_none_-s_activate
Instance	Word2vec 5
Best Features	feature_8
	feature_14
	feature_12
	feature_15
	feature_0
	feature_11
	feature_3
	feature_2

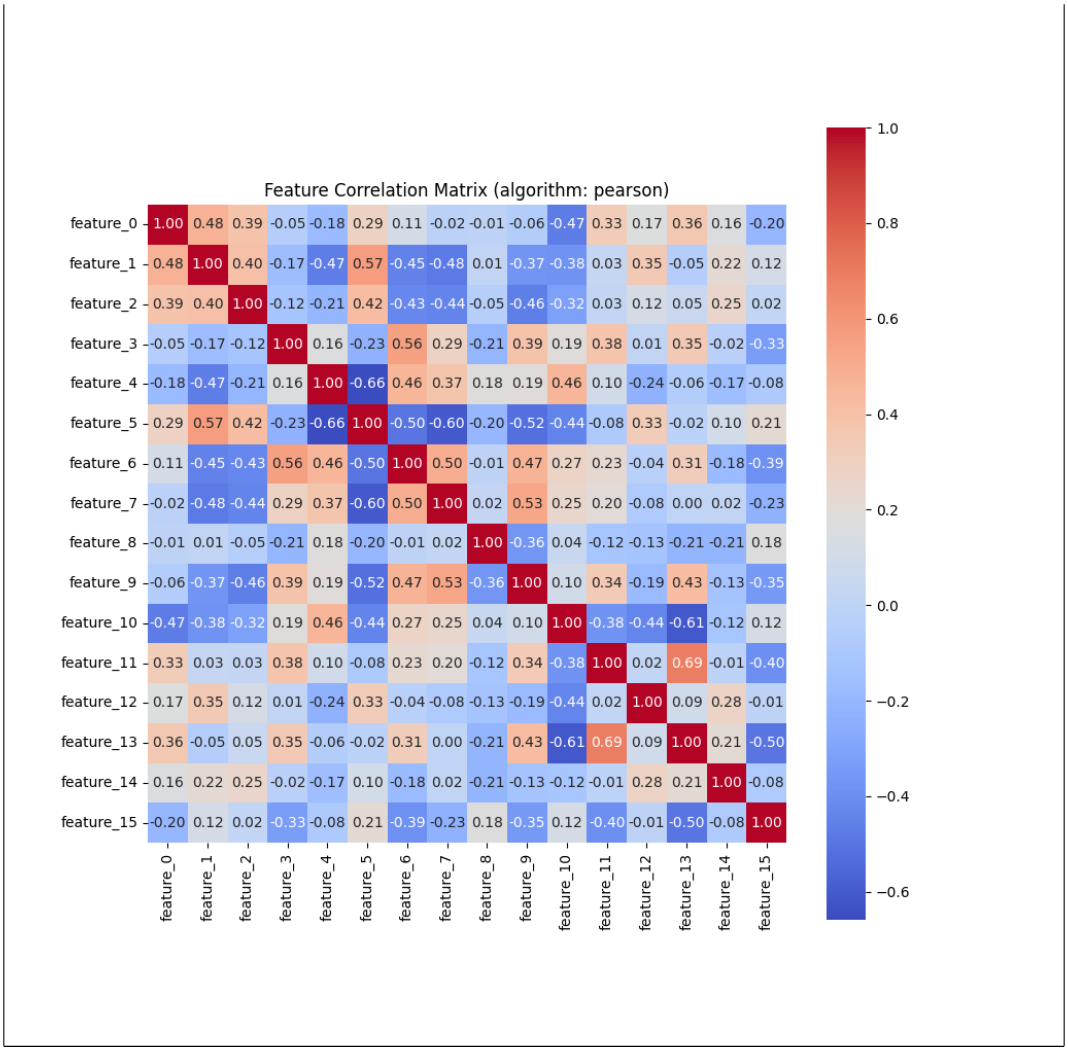


Table C.8: Word2vec 6 Feature Engineering Results on 25_-filtered_chunk_extraction_-e_none_-s_activate

Dataset Name	25_filtered_chunk_extraction_-e_none_-s_activate
Instance	Word2vec 6
Best Features	feature_13
	feature_1
	feature_9
	feature_3
	feature_10
	feature_7
	feature_8
	feature_4

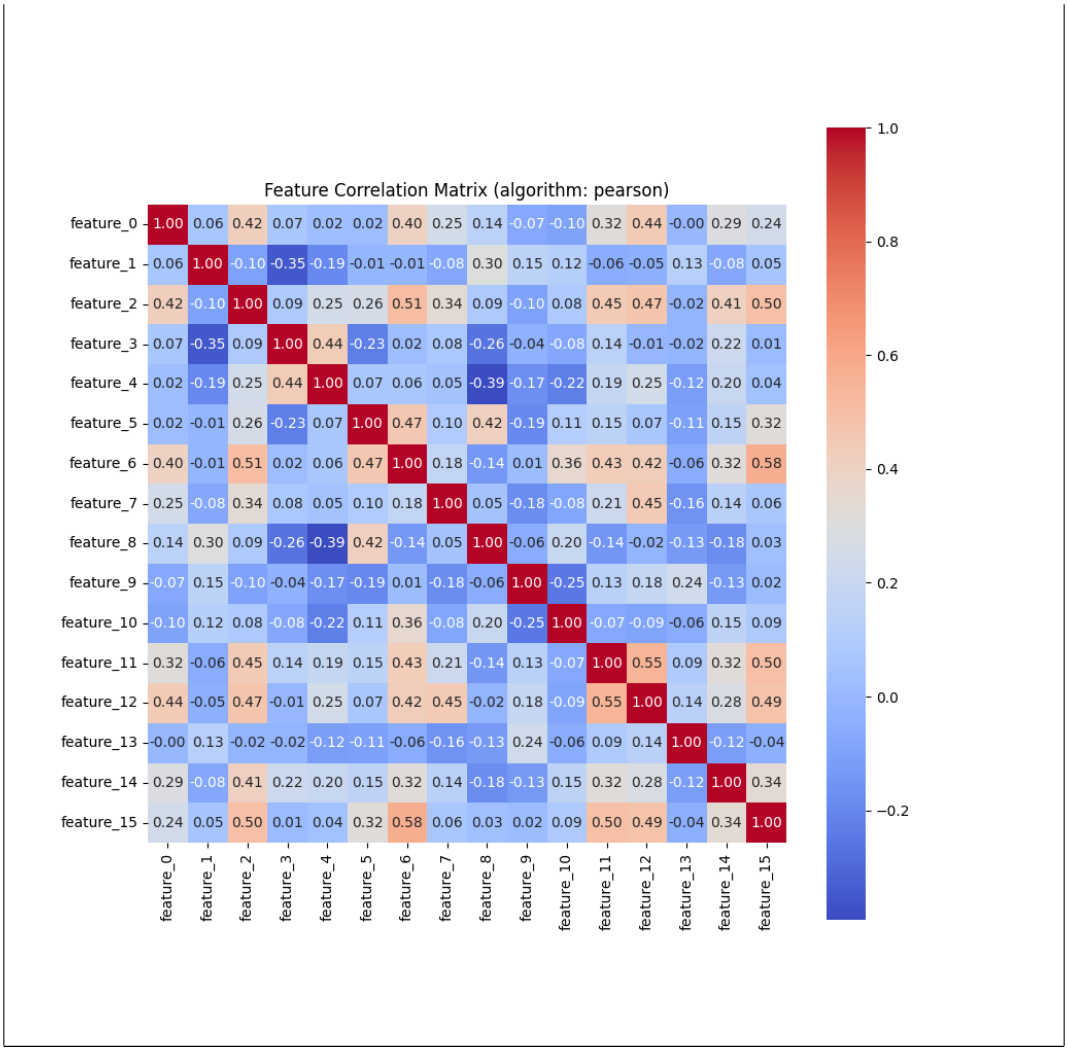


Table C.9: Word2vec 7 Feature Engineering Results on 25_-filtered_chunk_extraction_-e_none_-s_activate

Dataset Name	25_filtered_chunk_extraction_-e_none_-s_activate
Instance	Word2vec 7
Best Features	feature_2
	feature_15
	feature_8
	feature_14
	feature_10
	feature_1
	feature_0
	feature_13

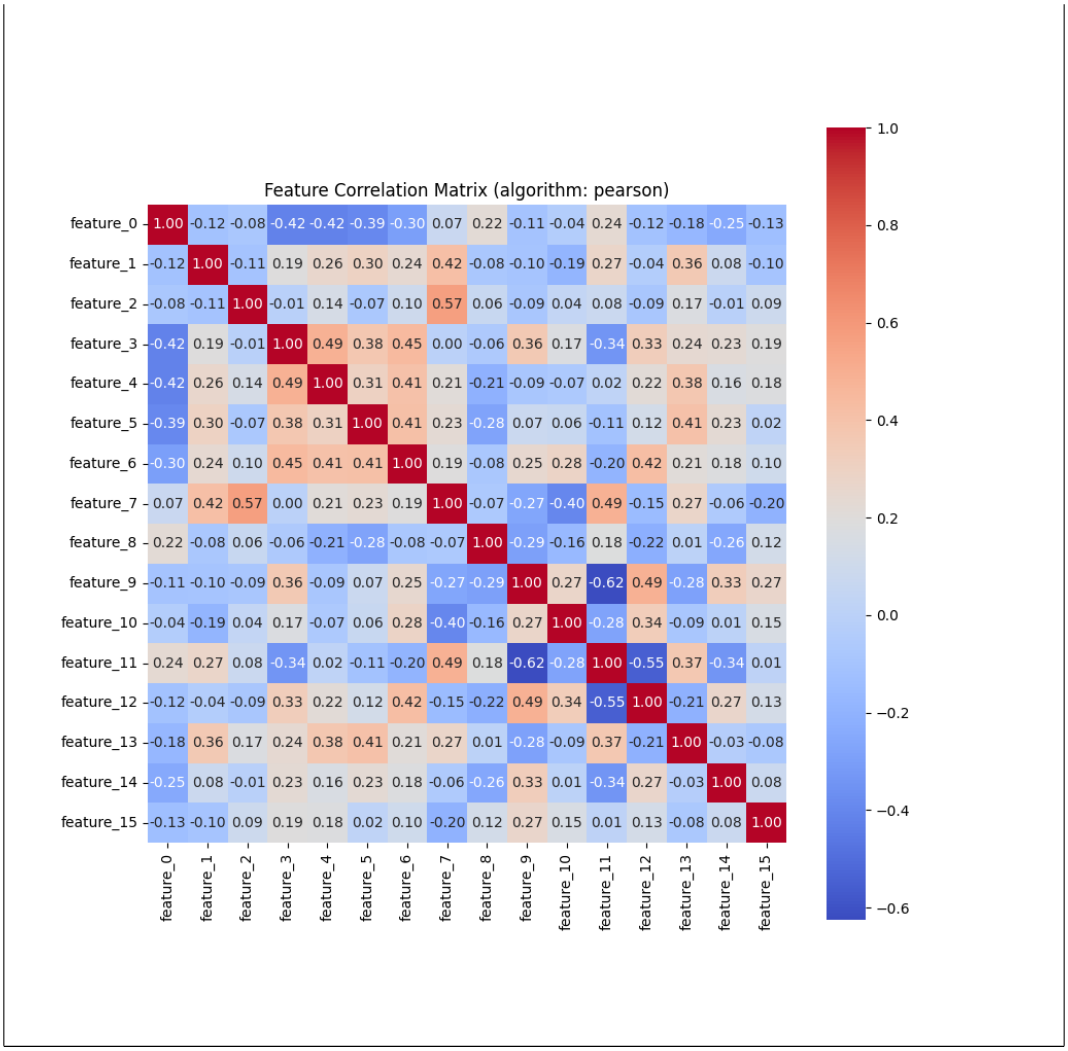


Table C.10: Word2vec 8 Feature Engineering Results on 25_-filtered_chunk_extraction_-e_none_-s_activate

Dataset Name	25_filtered_chunk_extraction_-e_none_-s_activate
Instance	Word2vec 8
Best Features	feature_3
	feature_31
	feature_62
	feature_78
	feature_93
	feature_11
	feature_32
	feature_61

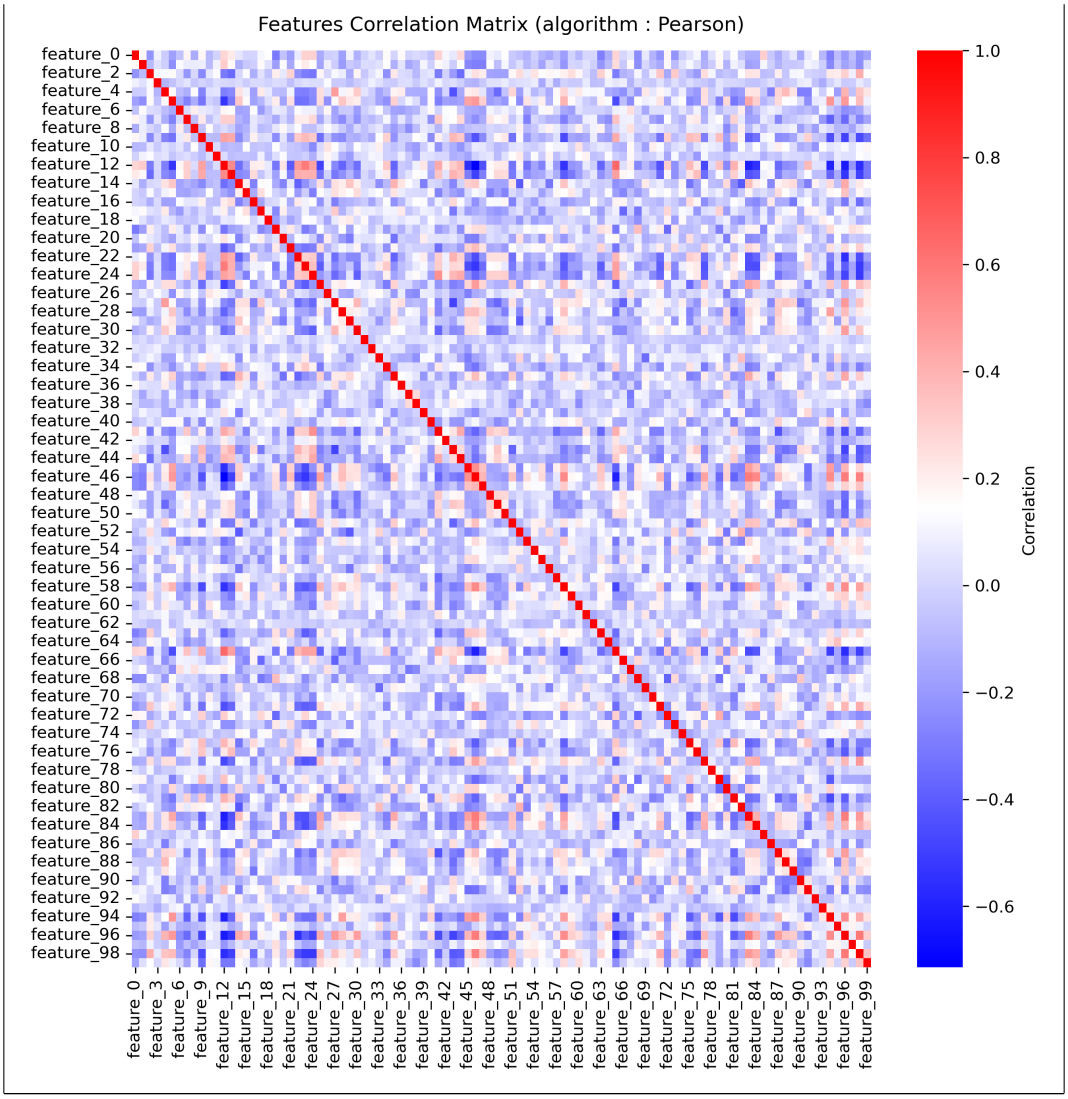
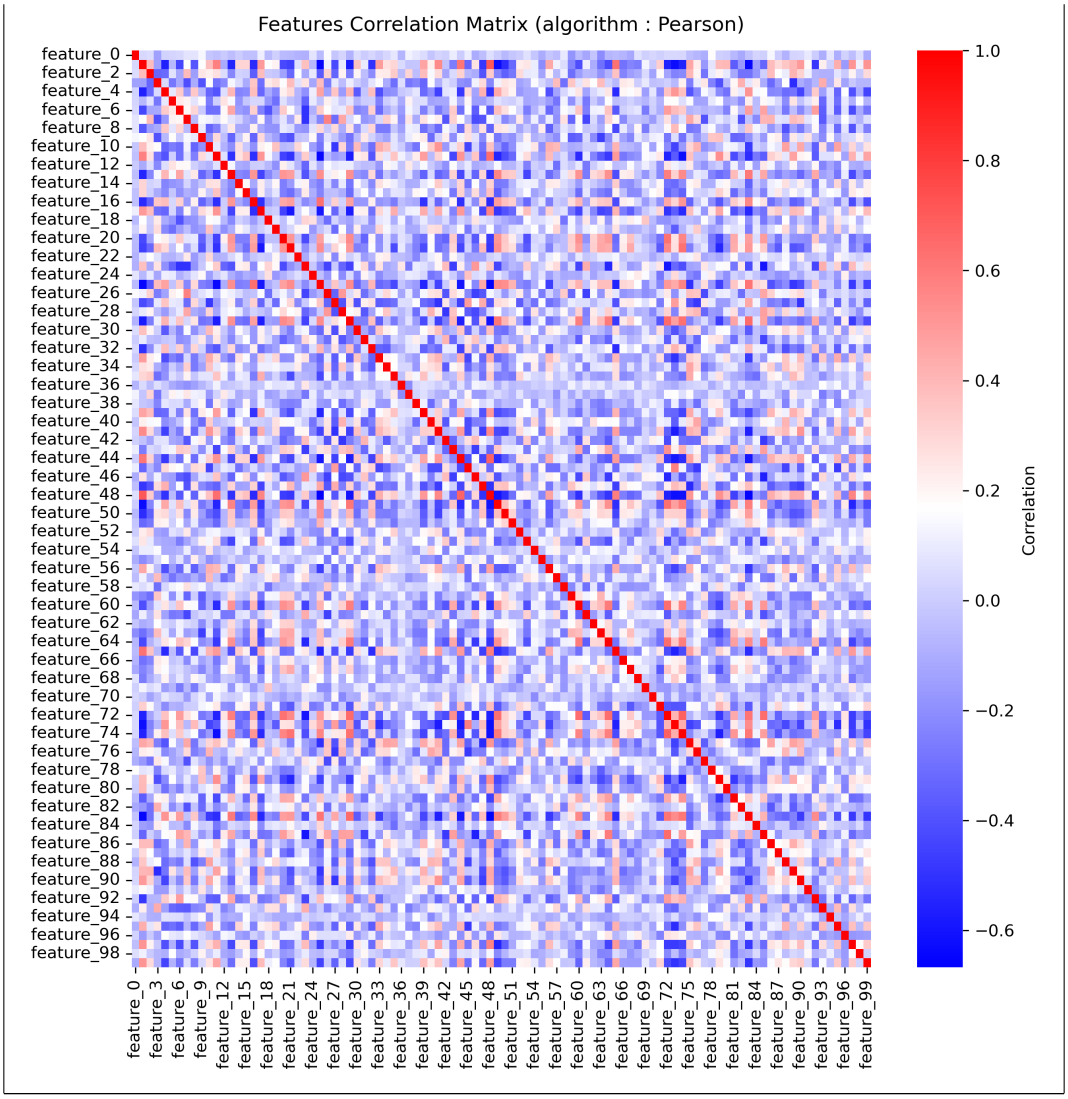


Table C.11: Word2vec 9 Feature Engineering Results on 25_-filtered_chunk_extraction_-e_none_-s_activate

Dataset Name	25_filtered_chunk_extraction_-e_none_-s_activate
Instance	Word2vec 9
Best Features	feature_0
	feature_36
	feature_94
	feature_70
	feature_69
	feature_22
	feature_84
	feature_37



C.3.2 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Table C.12: Transformers 0 Feature Engineering Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Dataset Name	26_filtered_chunk_extraction_-e_only-max-entropy_-s_none
Instance	Transformers 0
Best Features	embedded_6
	embedded_3
	embedded_0
	embedded_5
	embedded_4
	embedded_1
	embedded_2
	embedded_7

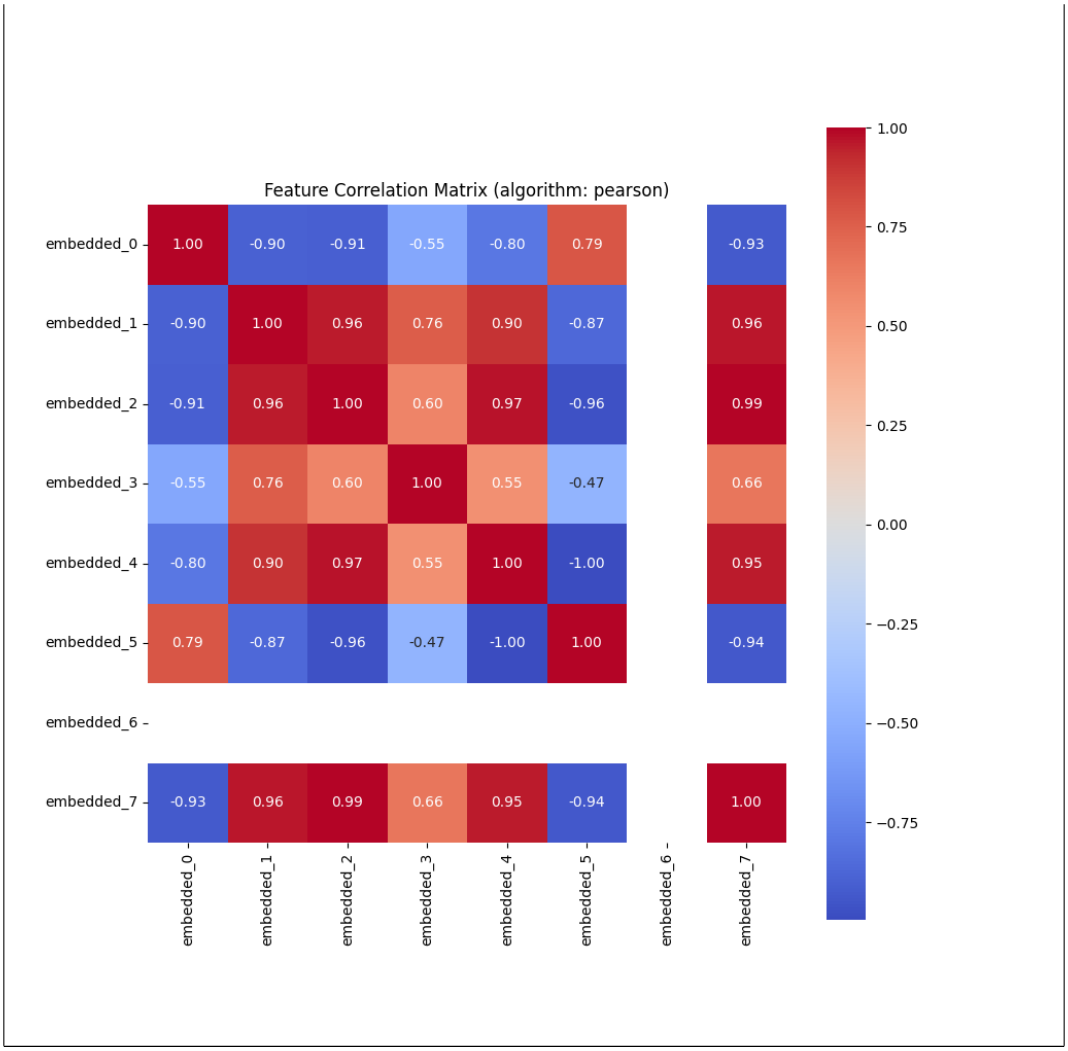


Table C.13: Transformers 1 Feature Engineering Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Dataset Name	26_filtered_chunk_extraction_-e_only-max-entropy_-s_none
Instance	Transformers 1
Best Features	embedded_2
	embedded_8
	embedded_14
	embedded_5
	embedded_6
	embedded_1
	embedded_13
	embedded_15

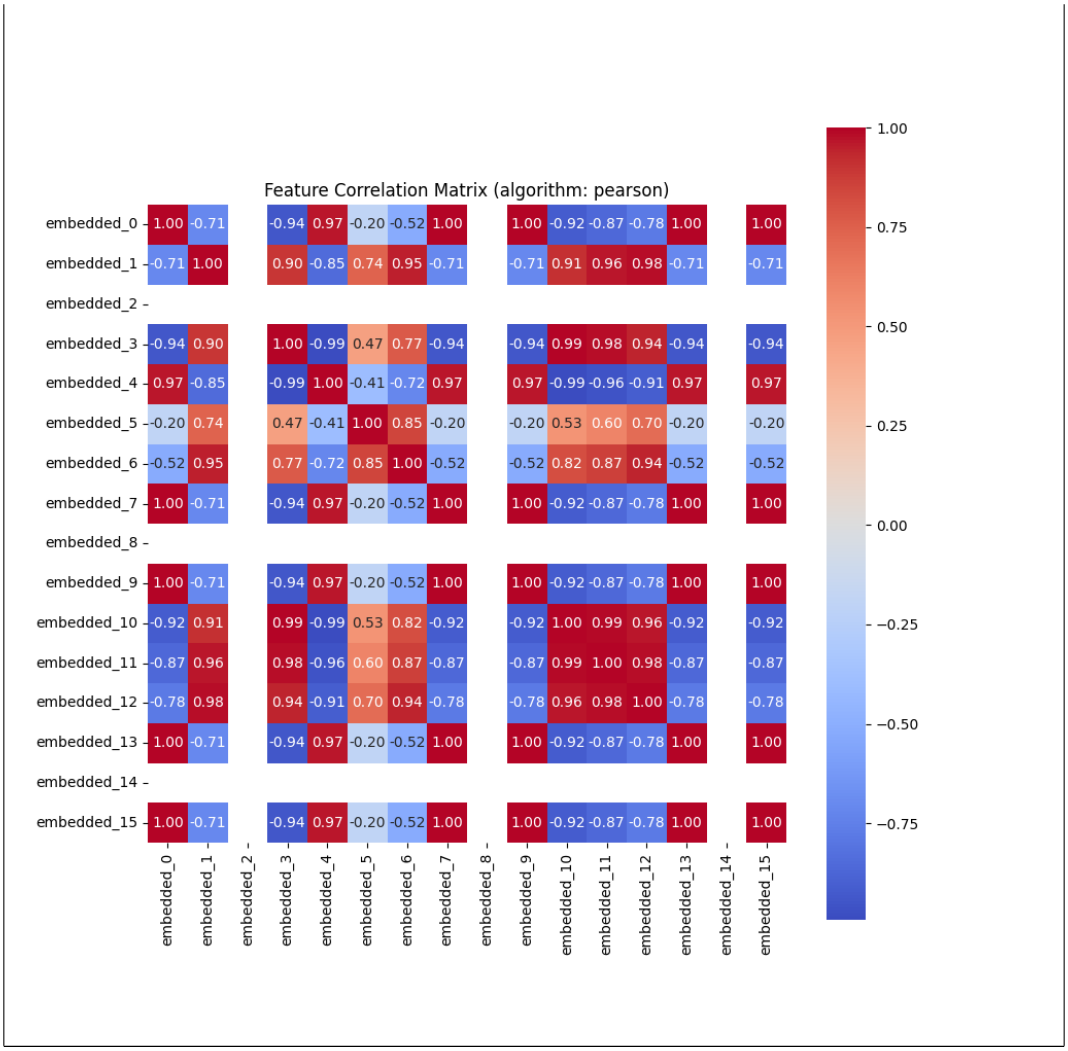


Table C.14: Word2vec 0 Feature Engineering Results on 26_-filtered_chunk_extraction_-e_only-max-entropy_-s_none

Dataset Name	26_filtered_chunk_extraction_-e_only-max-entropy_-s_none
Instance	Word2vec 0
Best Features	feature_3
	feature_5
	feature_2
	feature_6
	feature_4
	feature_7
	feature_1
	feature_0

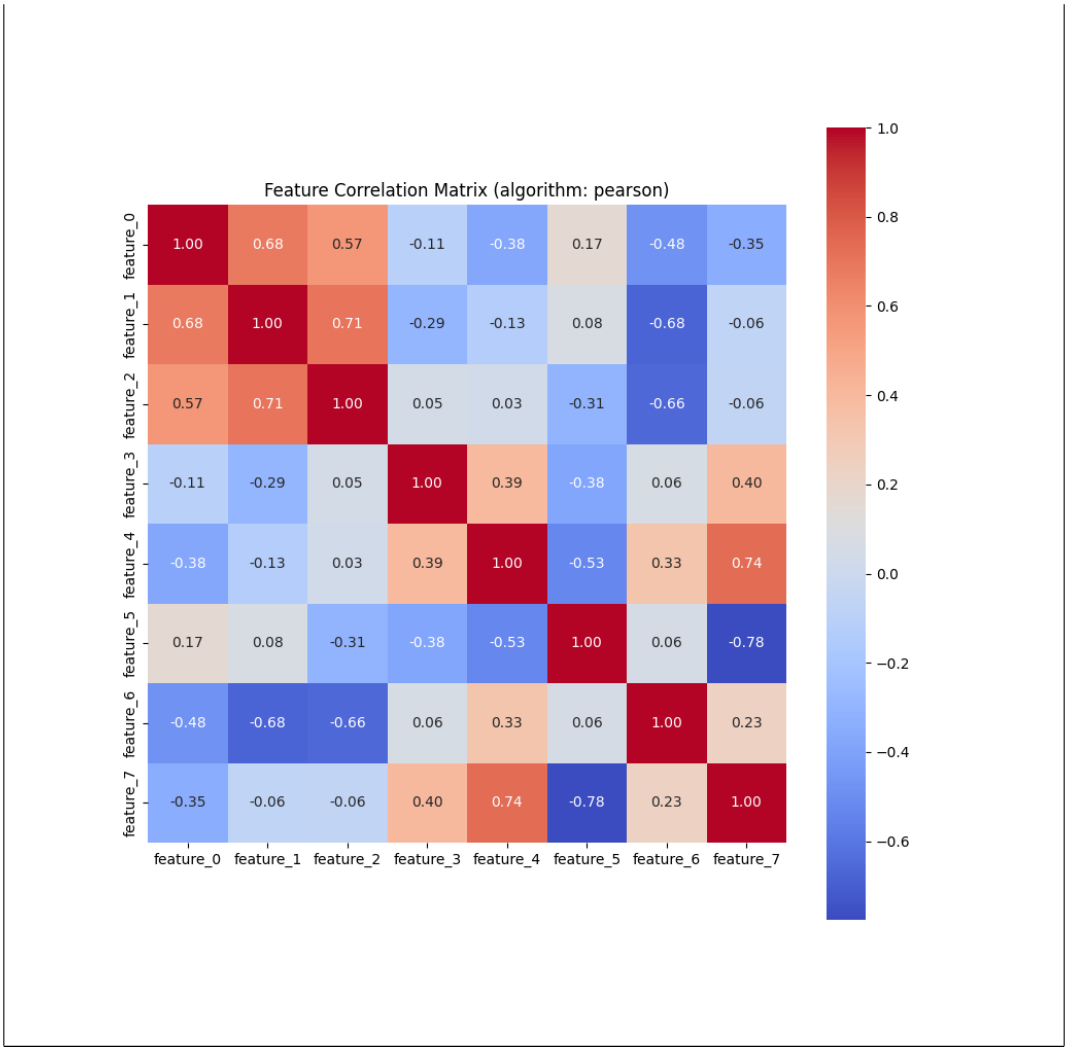


Table C.15: Word2vec 1 Feature Engineering Results on 26_-filtered_chunk_extraction_-e_only-max-entropy_-s_none

Dataset Name	26_filtered_chunk_extraction_-e_only-max-entropy_-s_none
Instance	Word2vec 1
Best Features	feature_4
	feature_5
	feature_6
	feature_7
	feature_2
	feature_3
	feature_0
	feature_1

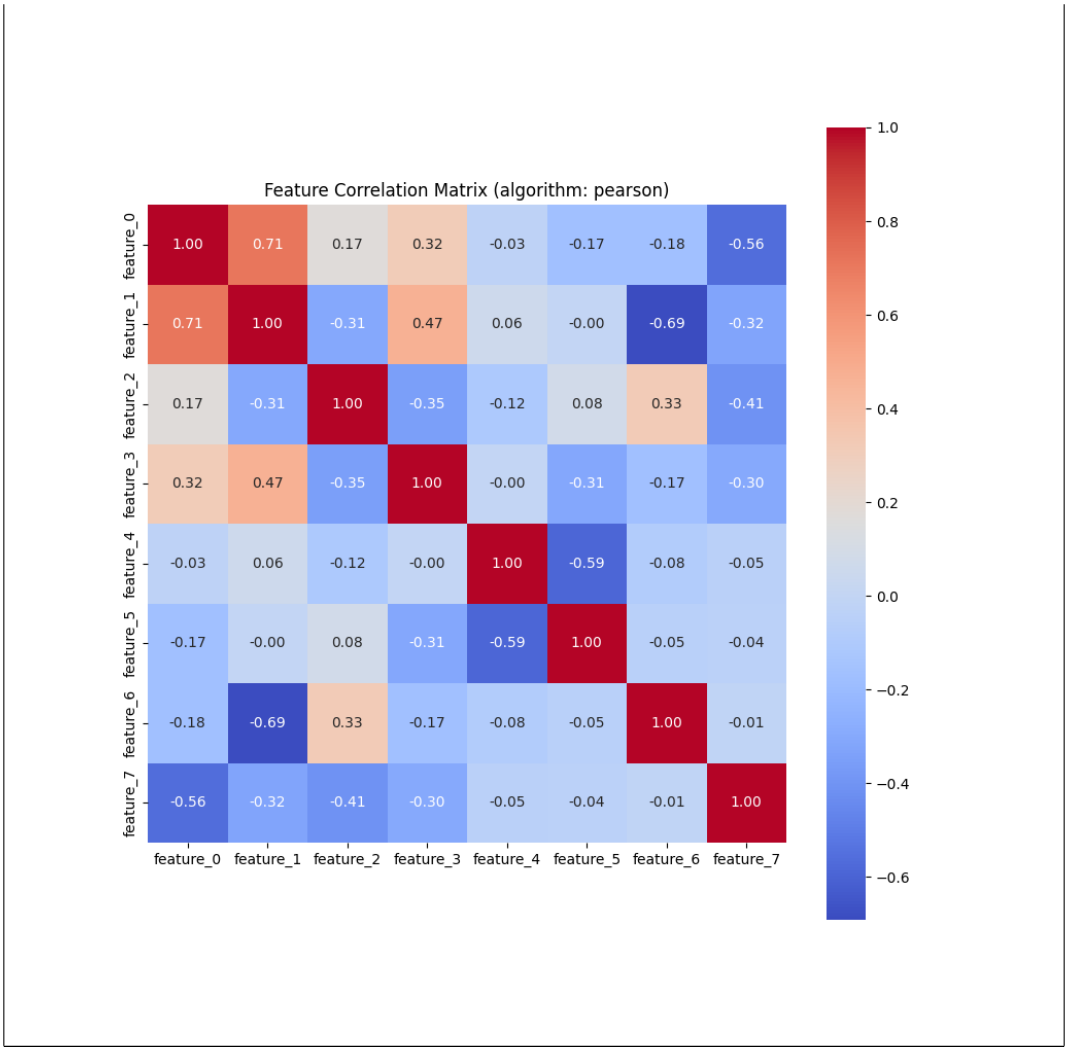


Table C.16: Word2vec 2 Feature Engineering Results on 26_-filtered_chunk_extraction_-e_only-max-entropy_-s_none

Dataset Name	26_filtered_chunk_extraction_-e_only-max-entropy_-s_none
Instance	Word2vec 2
Best Features	feature_2
	feature_0
	feature_3
	feature_6
	feature_5
	feature_4
	feature_7
	feature_1

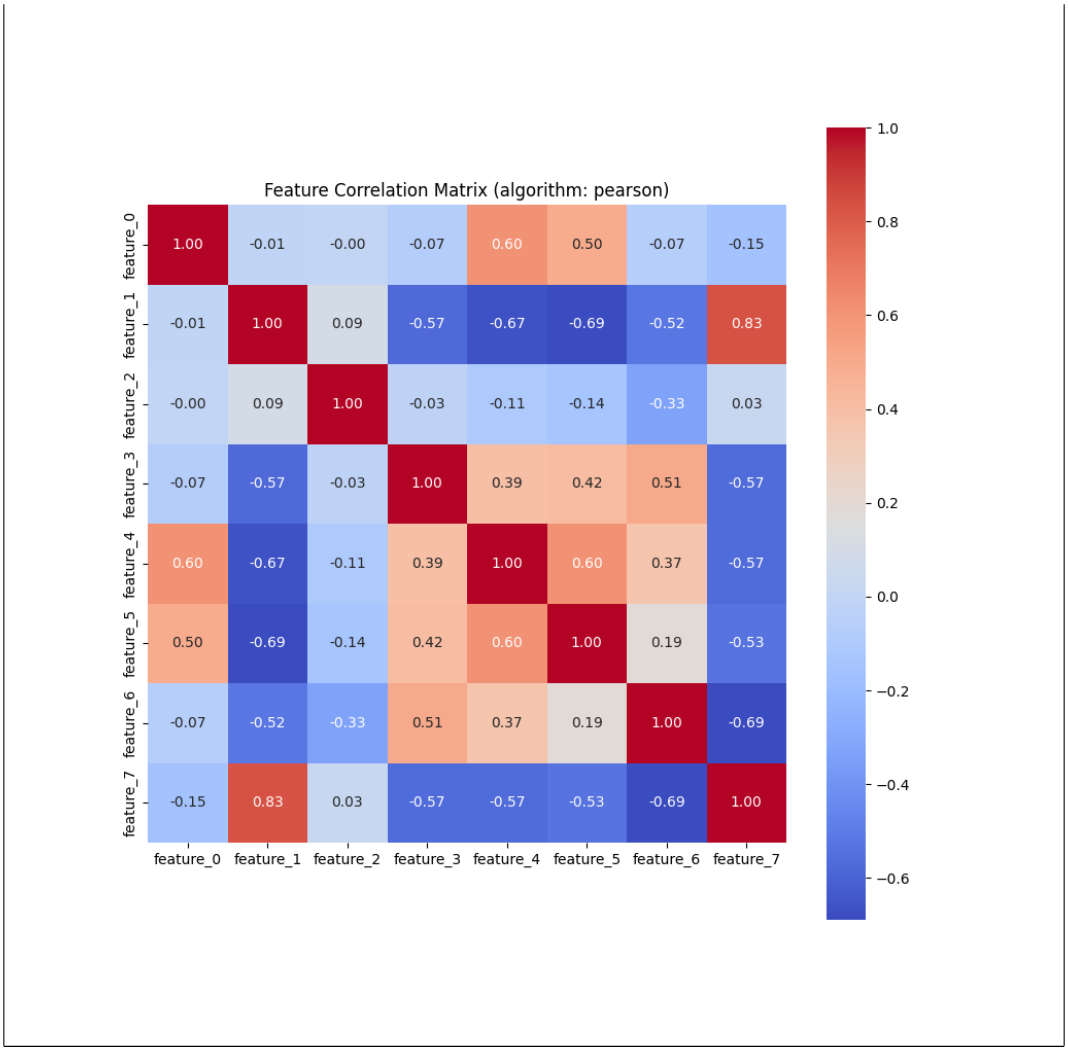


Table C.17: Word2vec 3 Feature Engineering Results on 26_-filtered_chunk_extraction_-e_only-max-entropy_-s_none

Dataset Name	26_filtered_chunk_extraction_-e_only-max-entropy_-s_none
Instance	Word2vec 3
Best Features	feature_4
	feature_2
	feature_7
	feature_0
	feature_5
	feature_6
	feature_3
	feature_1

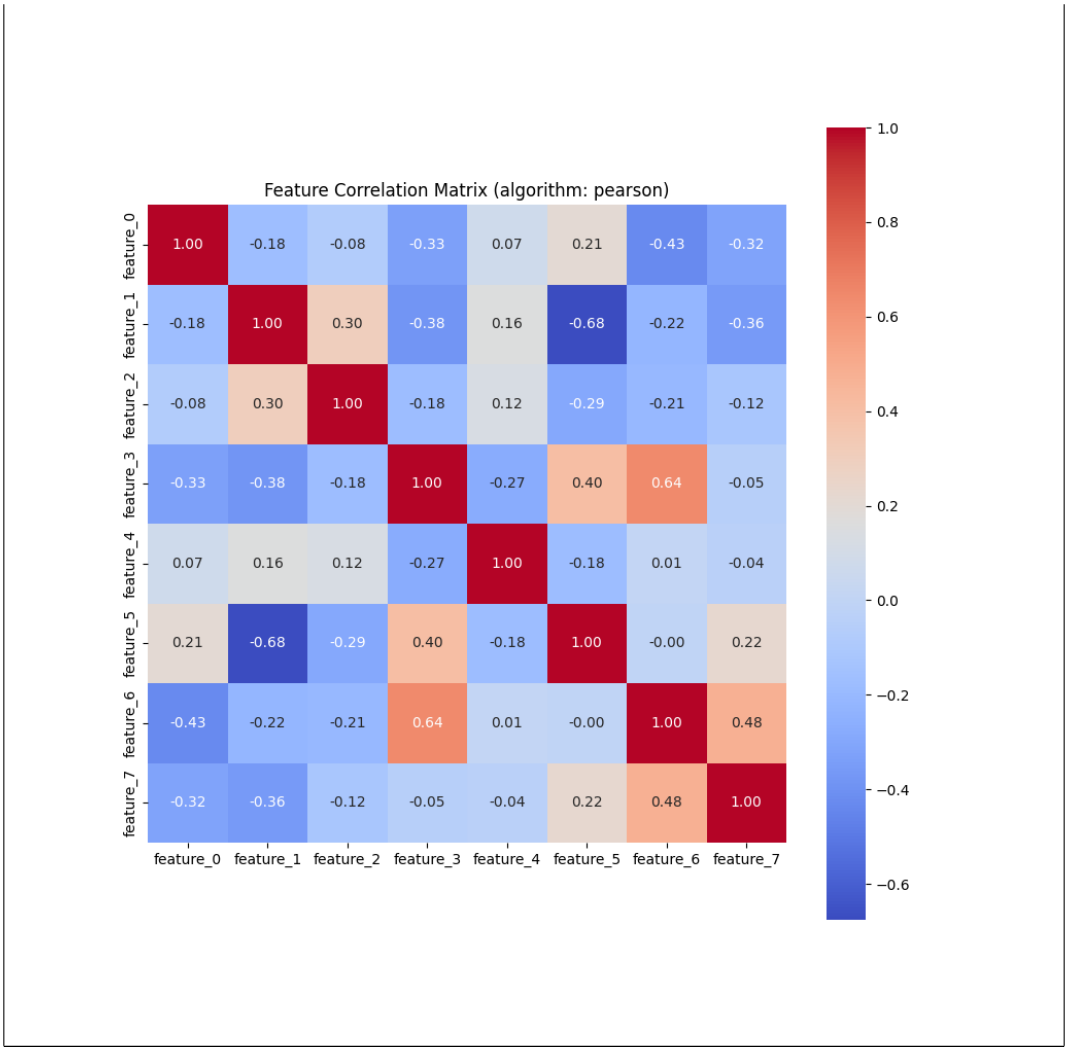


Table C.18: Word2vec 4 Feature Engineering Results on 26_-filtered_chunk_extraction_-e_only-max-entropy_-s_none

Dataset Name	26_filtered_chunk_extraction_-e_only-max-entropy_-s_none
Instance	Word2vec 4
Best Features	feature_10
	feature_0
	feature_5
	feature_12
	feature_11
	feature_7
	feature_6
	feature_3

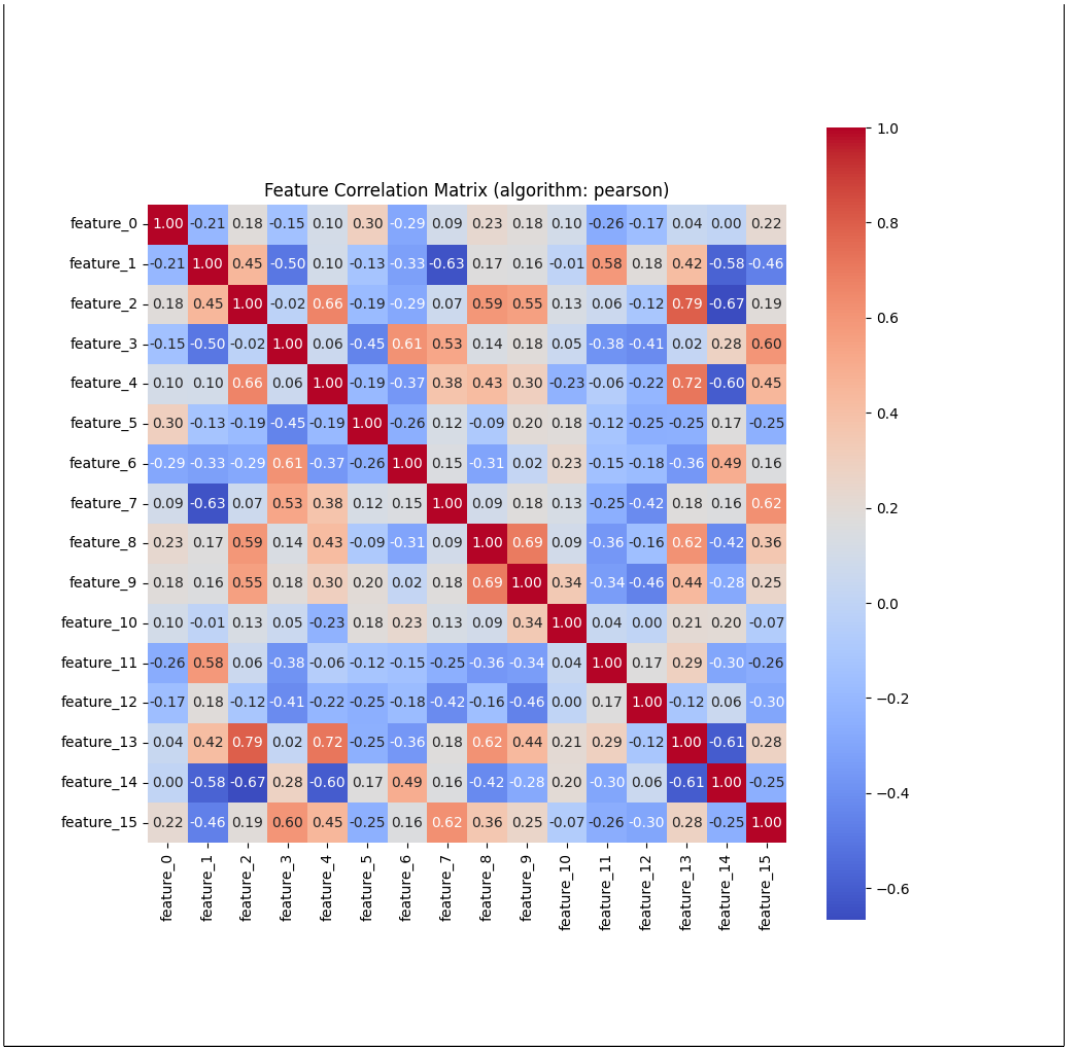


Table C.19: Word2vec 5 Feature Engineering Results on 26_-filtered_chunk_extraction_-e_only-max-entropy_-s_none

Dataset Name	26_filtered_chunk_extraction_-e_only-max-entropy_-s_none
Instance	Word2vec 5
Best Features	feature_3
	feature_10
	feature_8
	feature_5
	feature_9
	feature_12
	feature_15
	feature_2

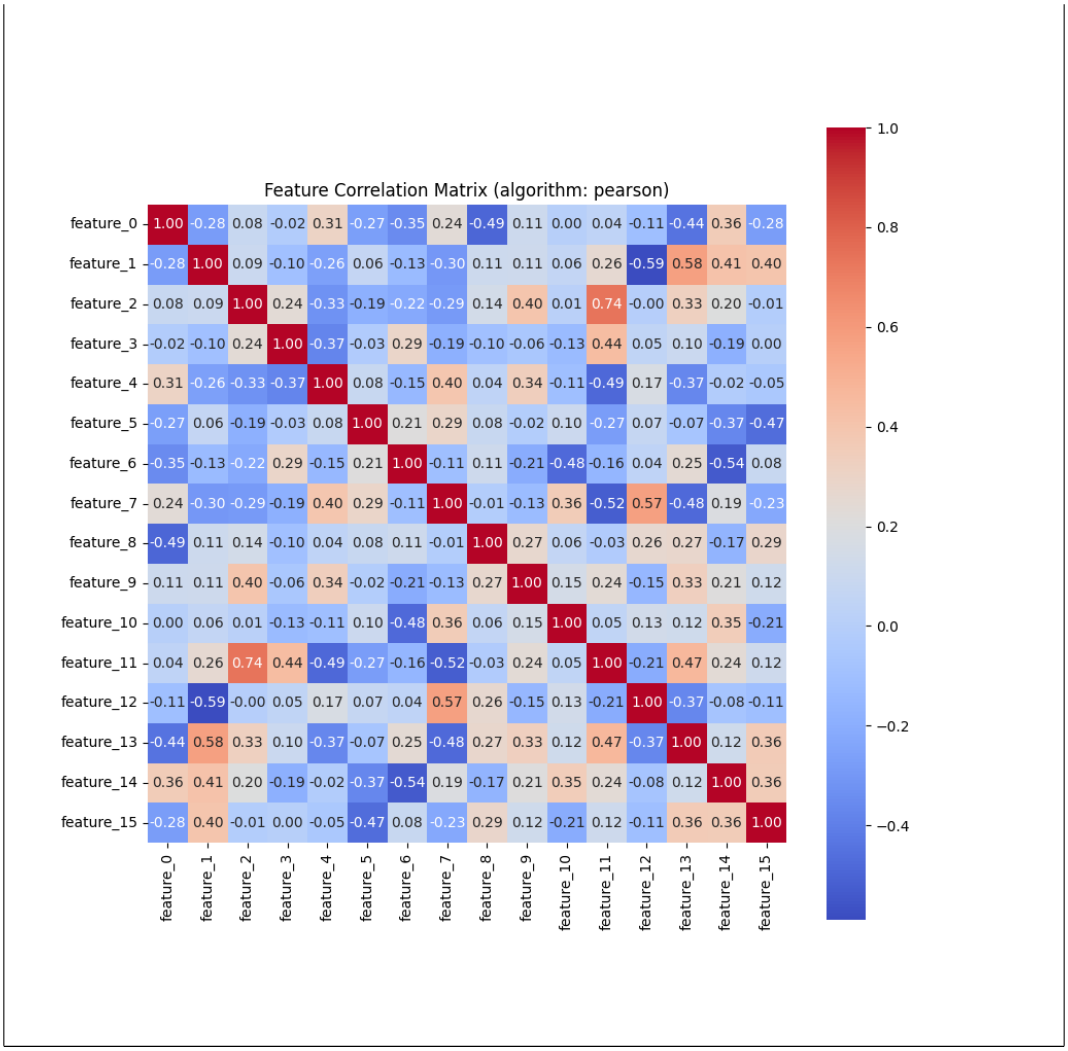


Table C.20: Word2vec 6 Feature Engineering Results on 26_-filtered_chunk_extraction_-e_only-max-entropy_-s_none

Dataset Name	26_filtered_chunk_extraction_-e_only-max-entropy_-s_none
Instance	Word2vec 6
Best Features	feature_13
	feature_1
	feature_5
	feature_2
	feature_3
	feature_9
	feature_6
	feature_4

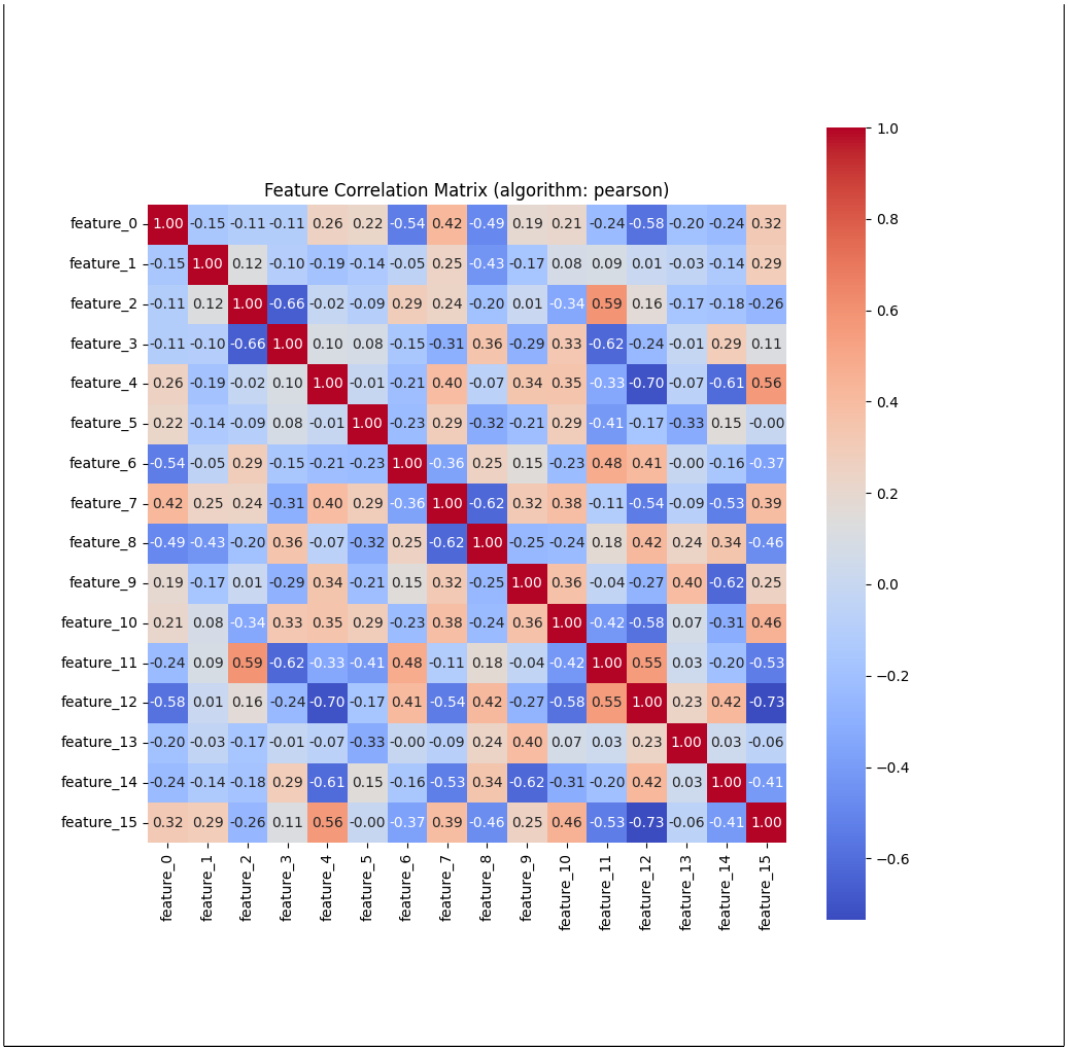


Table C.21: Word2vec 7 Feature Engineering Results on 26_-filtered_chunk_extraction_-e_only-max-entropy_-s_none

Dataset Name	26_filtered_chunk_extraction_-e_only-max-entropy_-s_none
Instance	Word2vec 7
Best Features	feature_4
	feature_3
	feature_12
	feature_10
	feature_15
	feature_0
	feature_7
	feature_9

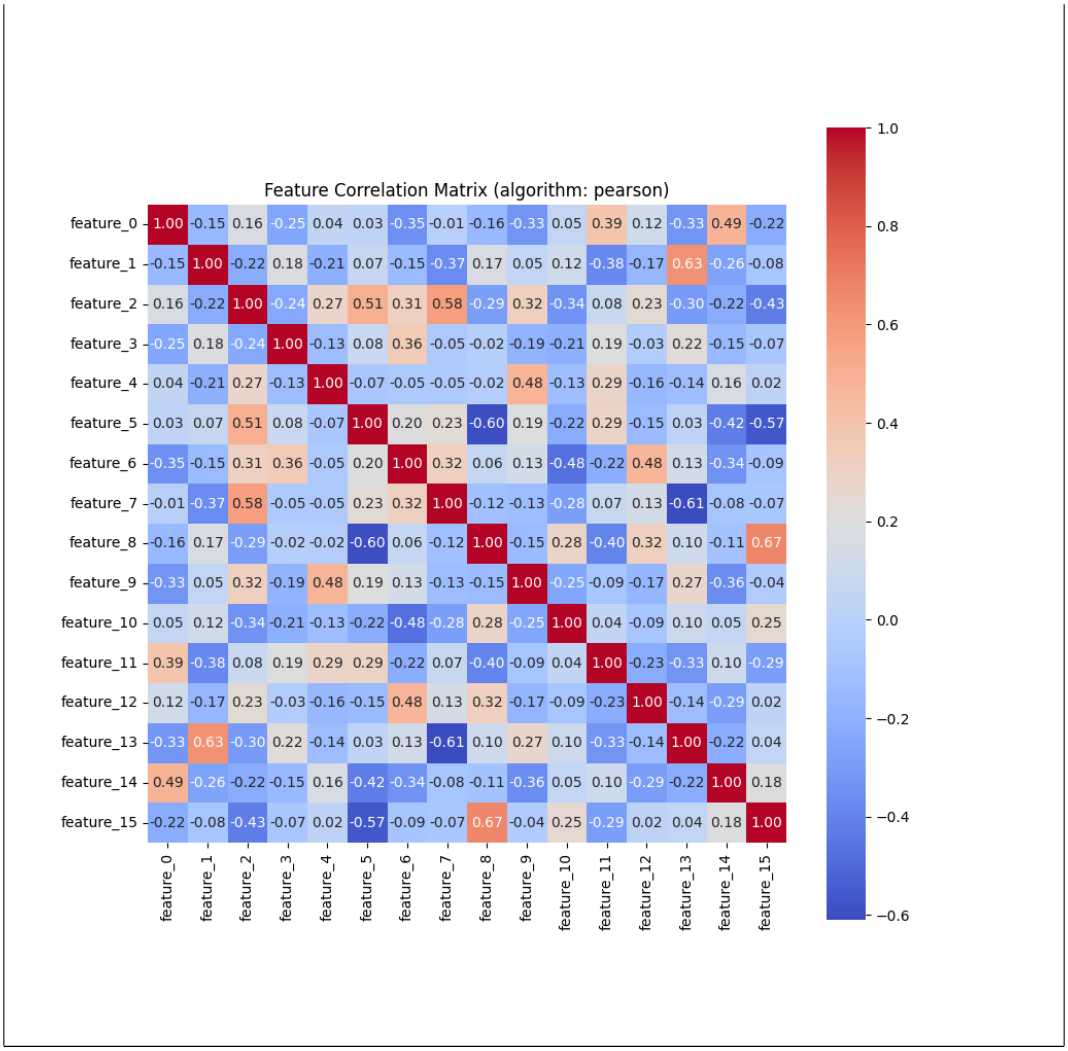


Table C.22: Word2vec 8 Feature Engineering Results on 26_-filtered_chunk_extraction_-e_only-max-entropy_-s_none

Dataset Name	26_filtered_chunk_extraction_-e_only-max-entropy_-s_none
Instance	Word2vec 8
Best Features	feature_60
	feature_80
	feature_55
	feature_85
	feature_16
	feature_75
	feature_31
	feature_51

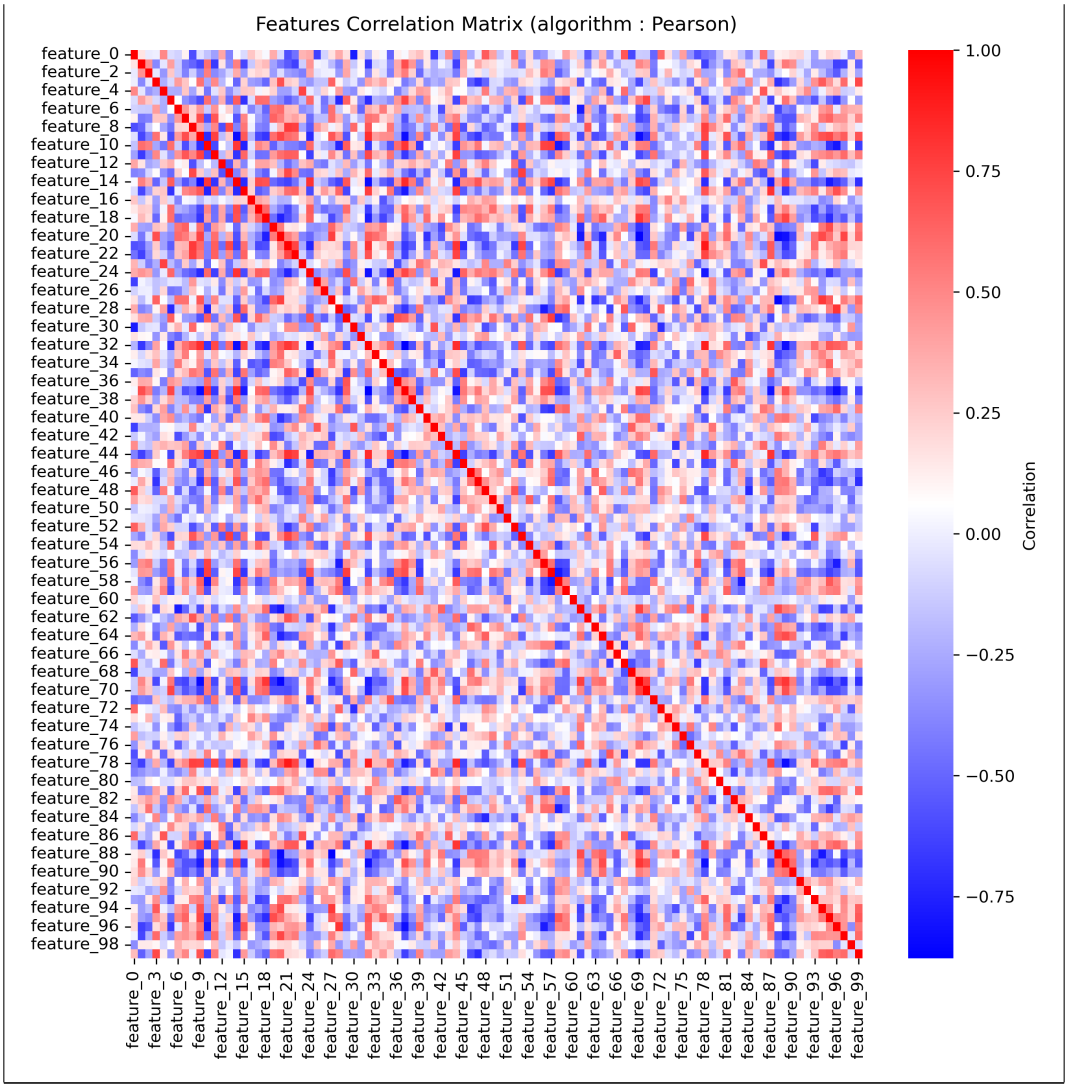


Table C.23: Word2vec 9 Feature Engineering Results on 26_-filtered_chunk_extraction_-e_only-max-entropy_-s_none

Dataset Name	26_filtered_chunk_extraction_-e_only-max-entropy_-s_none
Instance	Word2vec 9
Best Features	feature_34
	feature_40
	feature_19
	feature_30
	feature_28
	feature_62
	feature_35
	feature_91

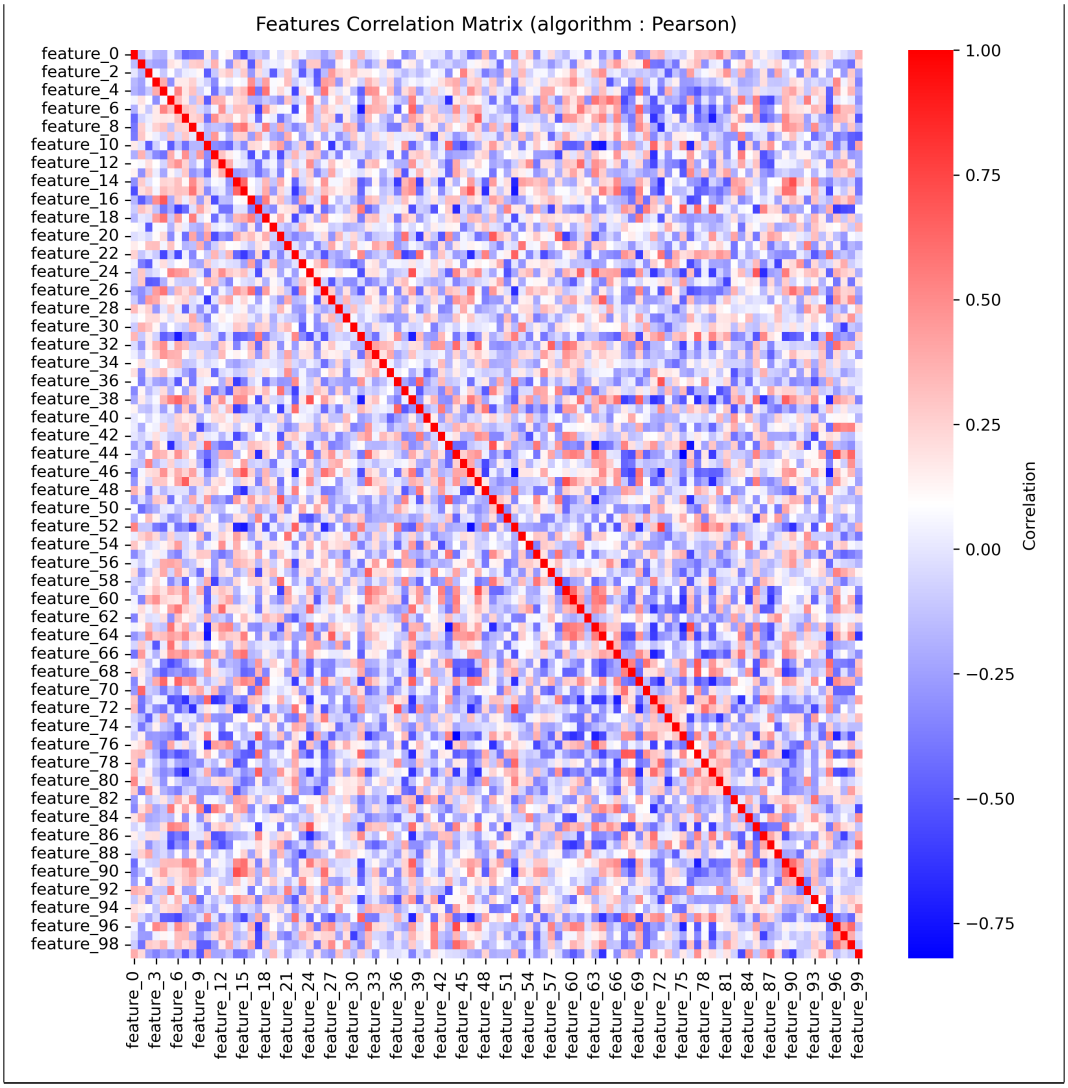


Table C.24: Word2vec 10 Feature Engineering Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Dataset Name	26_filtered_chunk_extraction_-e_only-max-entropy_-s_none
Instance	Word2vec 10
Best Features	feature_24
	feature_4
	feature_85
	feature_46
	feature_71
	feature_76
	feature_42
	feature_19

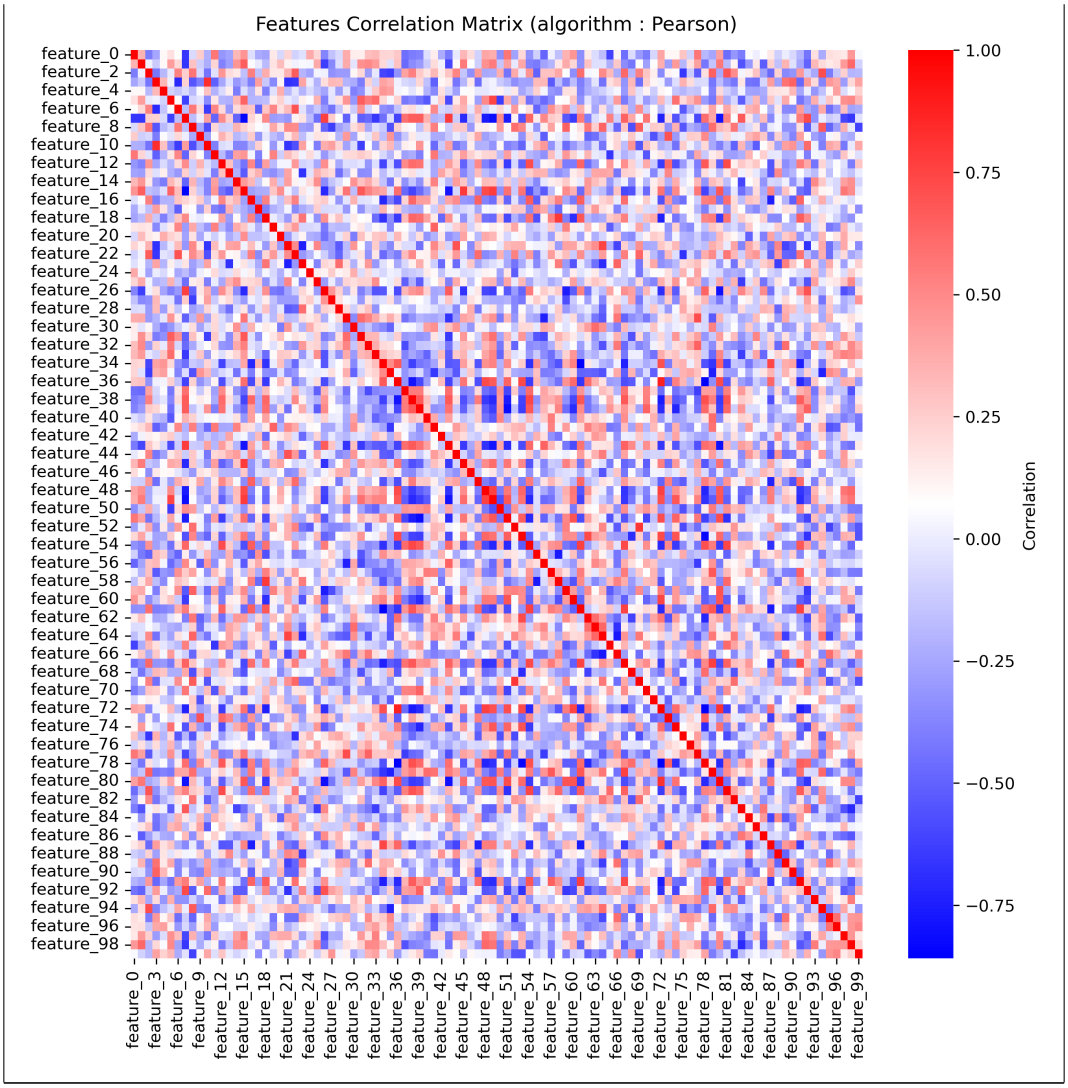
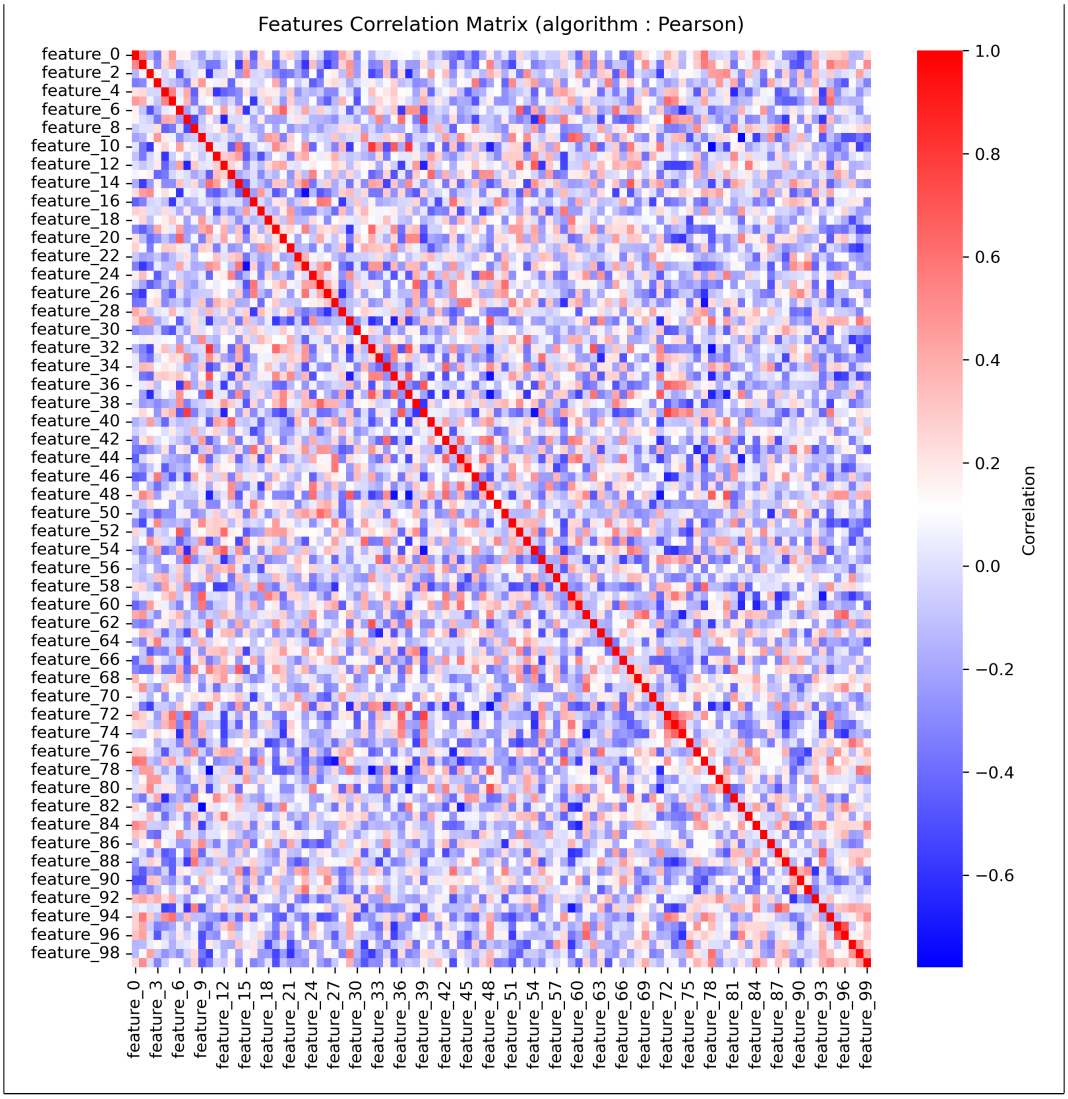


Table C.25: Word2vec 11 Feature Engineering Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Dataset Name	26_filtered_chunk_extraction_-e_only-max-entropy_-s_none
Instance	Word2vec 11
Best Features	feature_41
	feature_85
	feature_22
	feature_79
	feature_70
	feature_18
	feature_64
	feature_91



C.3.3 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Table C.26: Transformers 0 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Transformers 0
Best Features	embedded_0
	embedded_1
	embedded_4
	embedded_5
	embedded_7
	embedded_6
	embedded_3
	embedded_2

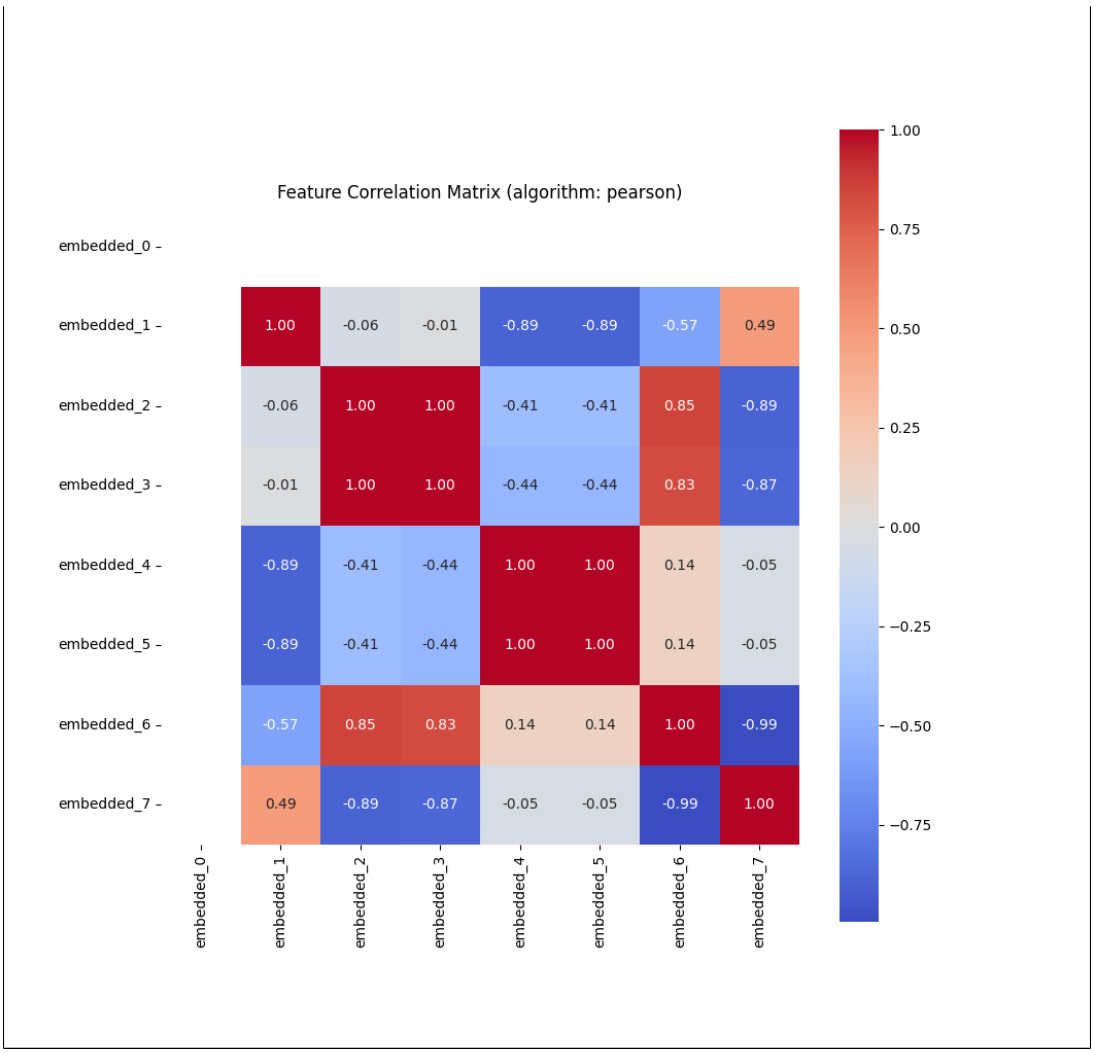


Table C.27: Transformers 1 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Transformers 1
Best Features	embedded_0
	embedded_6
	embedded_7
	embedded_9
	embedded_8
	embedded_10
	embedded_5
	embedded_3

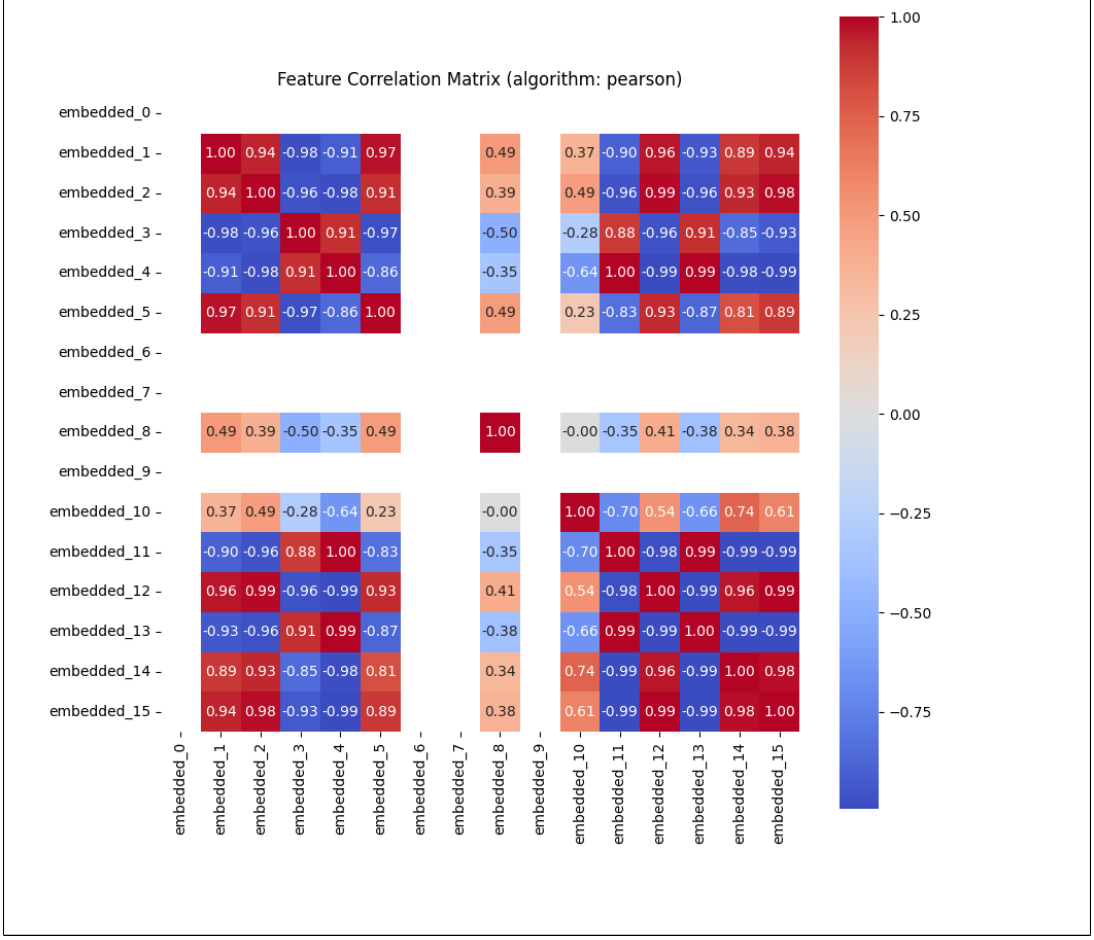


Table C.28: Transformers 2 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Transformers 2
Best Features	embedded_0
	embedded_3
	embedded_7
	embedded_6
	embedded_1
	embedded_2
	embedded_5
	embedded_4

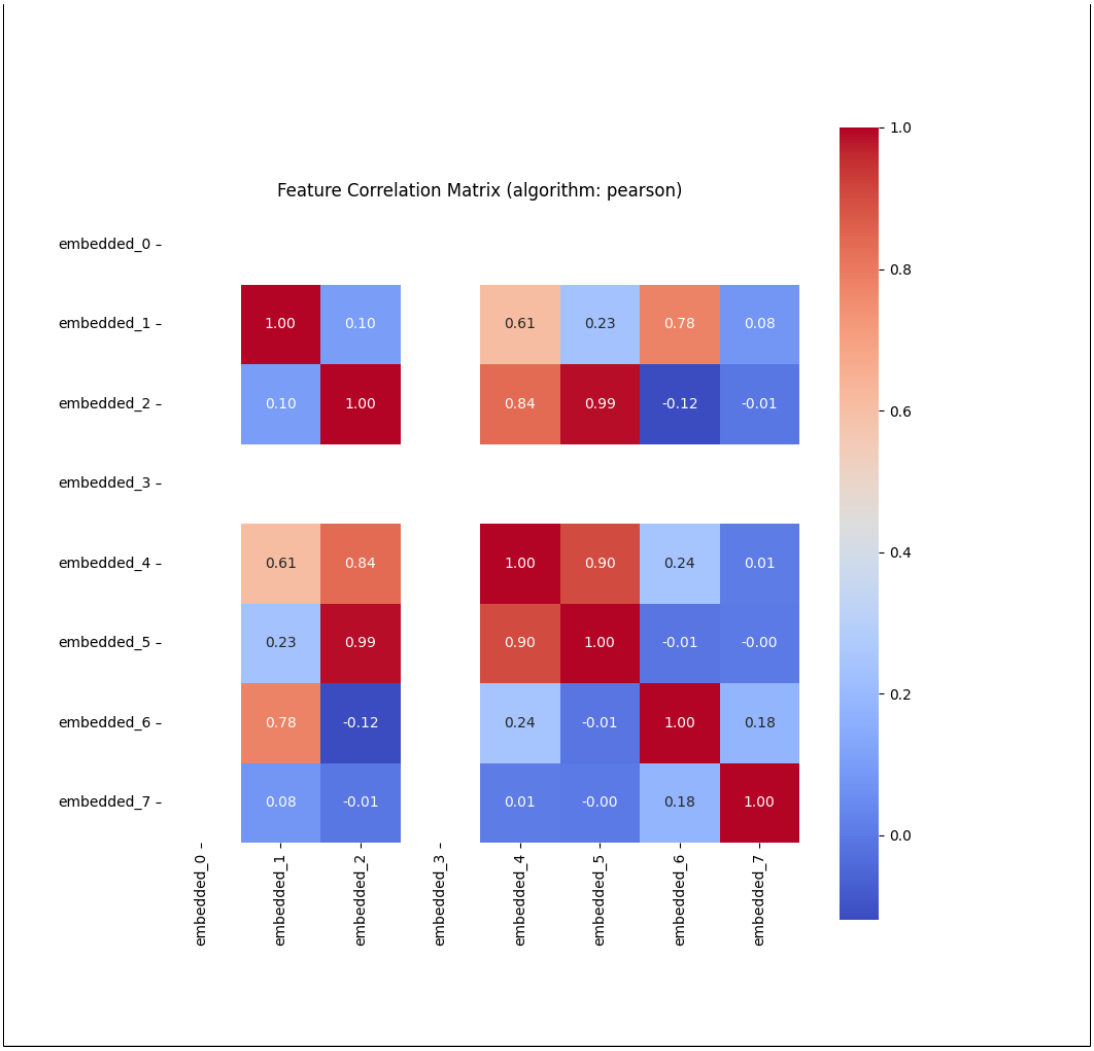


Table C.29: Transformers 3 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Transformers 3
Best Features	embedded_0
	embedded_2
	embedded_3
	embedded_7
	embedded_14
	embedded_9
	embedded_13
	embedded_5

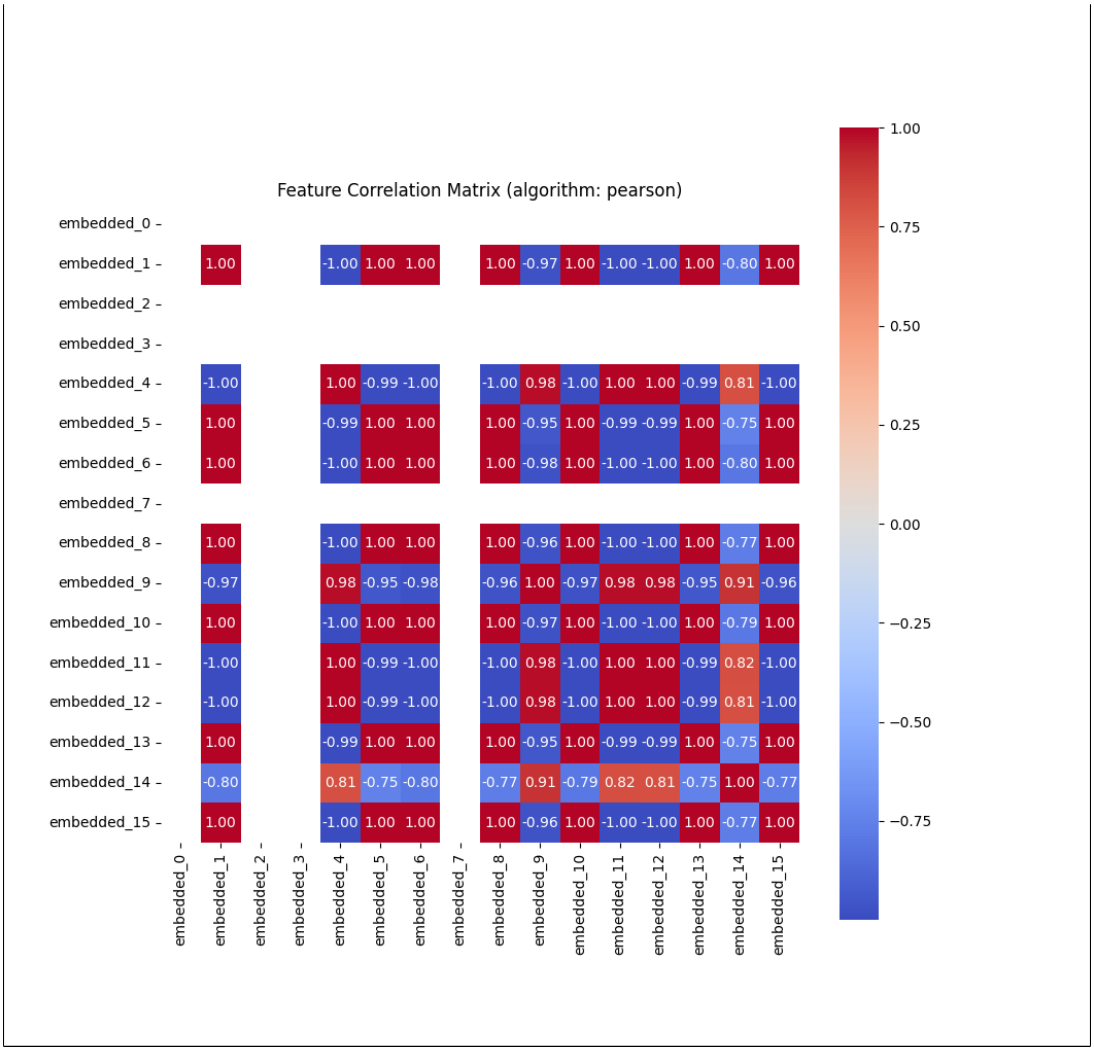


Table C.30: Transformers 4 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Transformers 4
Best Features	embedded_4
	embedded_1
	embedded_5
	embedded_7
	embedded_6
	embedded_3
	embedded_0
	embedded_2

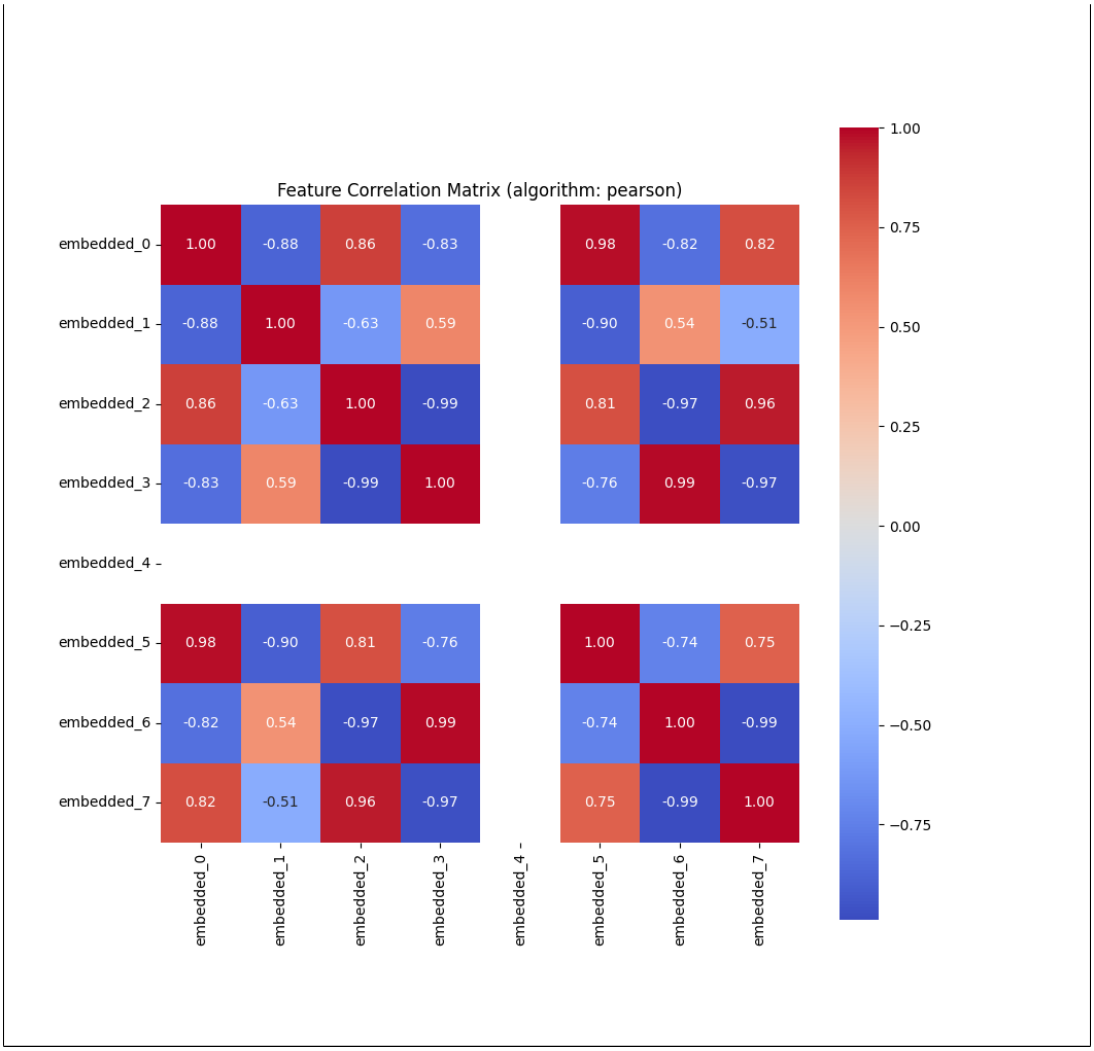


Table C.31: Transformers 5 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Transformers 5
Best Features	embedded_2
	embedded_4
	embedded_5
	embedded_6
	embedded_8
	embedded_14
	embedded_15
	embedded_0

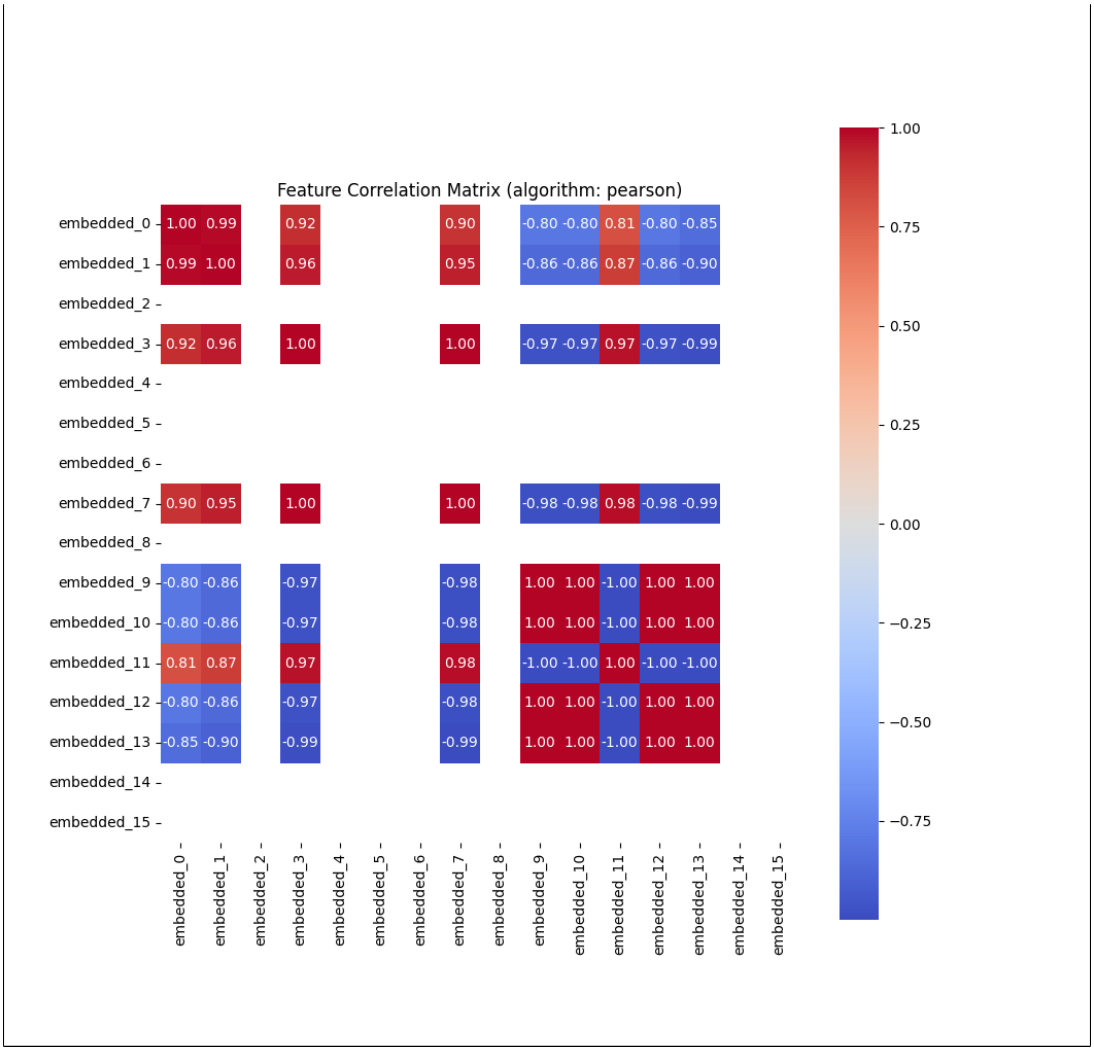


Table C.32: Transformers 6 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Transformers 6
Best Features	embedded_2
	embedded_3
	embedded_0
	embedded_5
	embedded_6
	embedded_1
	embedded_7
	embedded_4

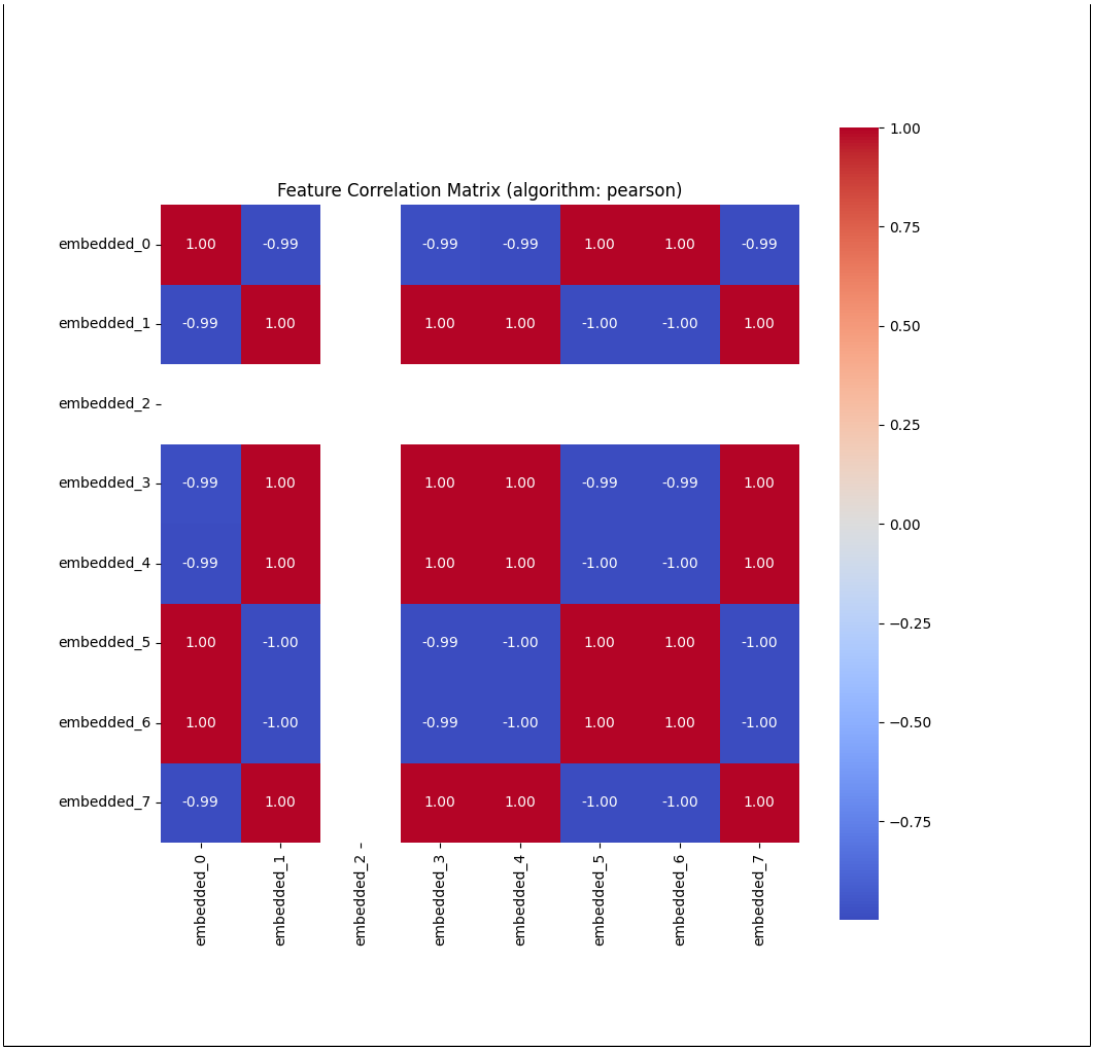


Table C.33: Transformers 7 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Transformers 7
Best Features	embedded_5
	embedded_11
	embedded_12
	embedded_13
	embedded_14
	embedded_15
	embedded_3
	embedded_4

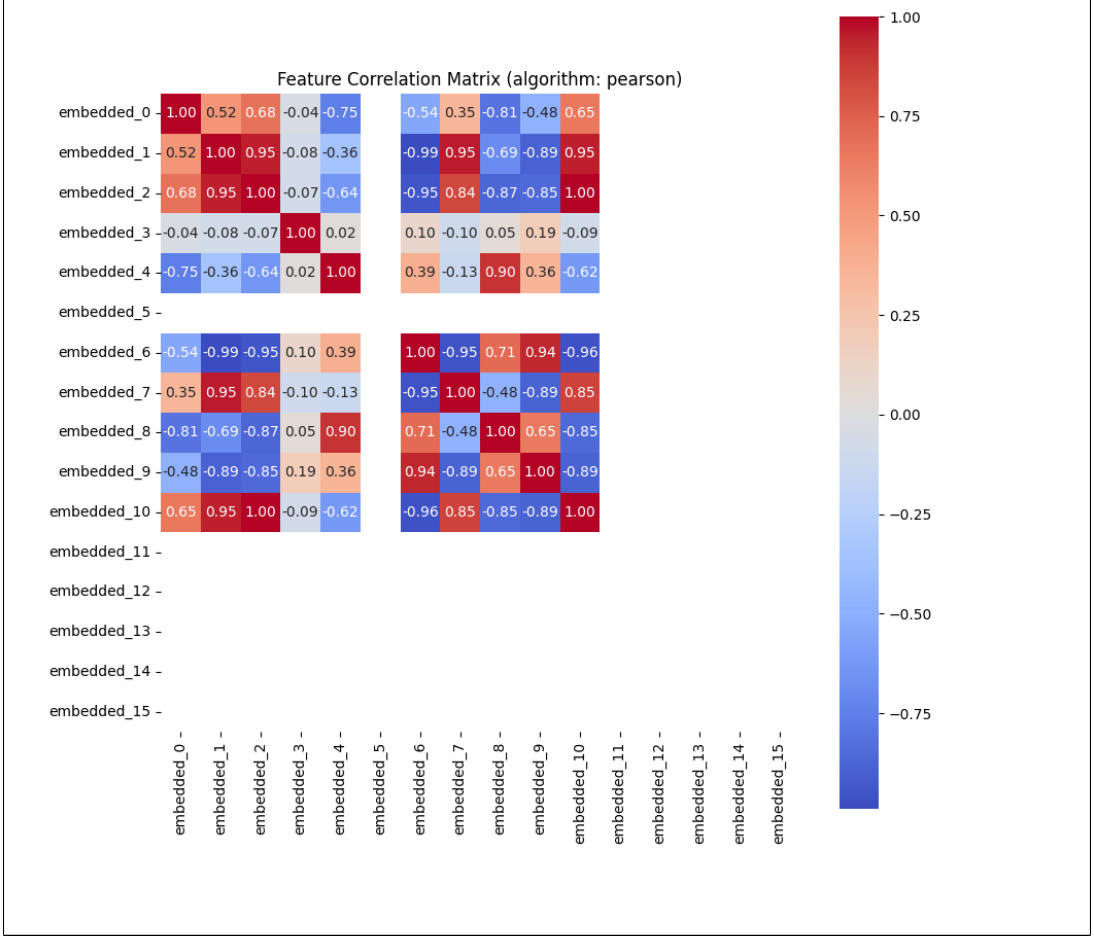


Table C.34: Word2vec 0 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Word2vec 0
Best Features	feature_1
	feature_6
	feature_4
	feature_7
	feature_5
	feature_2
	feature_3
	feature_0

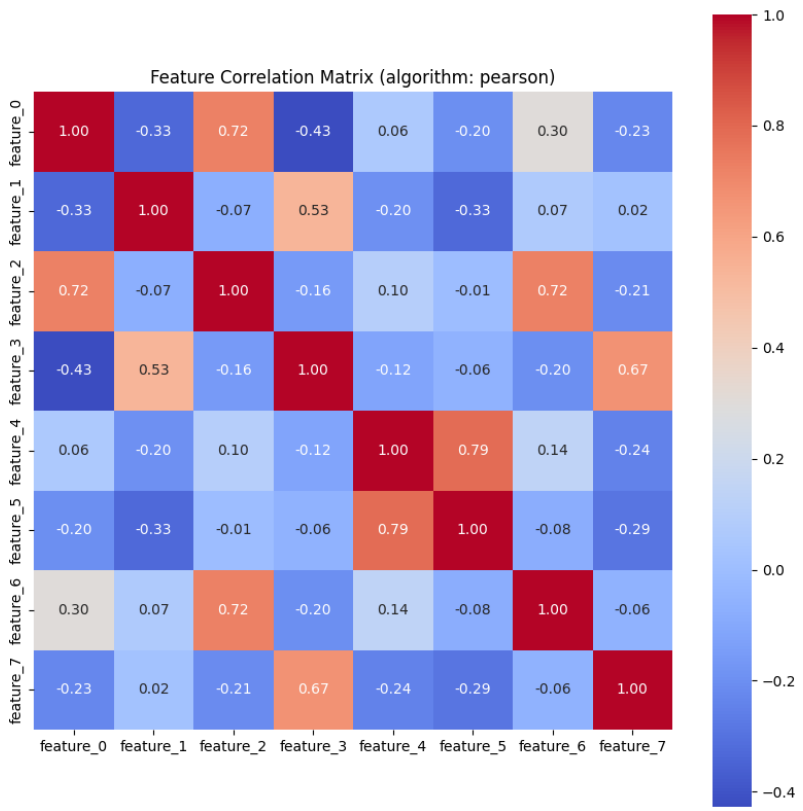


Table C.35: Word2vec 1 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Word2vec 1
Best Features	feature_3
	feature_2
	feature_6
	feature_4
	feature_0
	feature_7
	feature_1
	feature_5

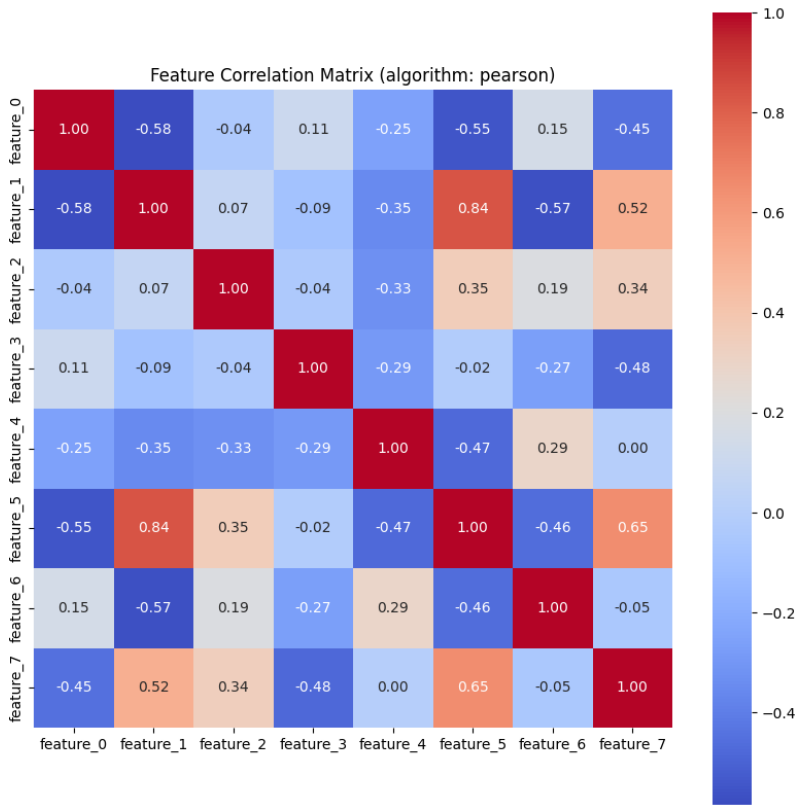


Table C.36: Word2vec 2 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Word2vec 2
Best Features	feature_2
	feature_7
	feature_1
	feature_4
	feature_5
	feature_6
	feature_3
	feature_0

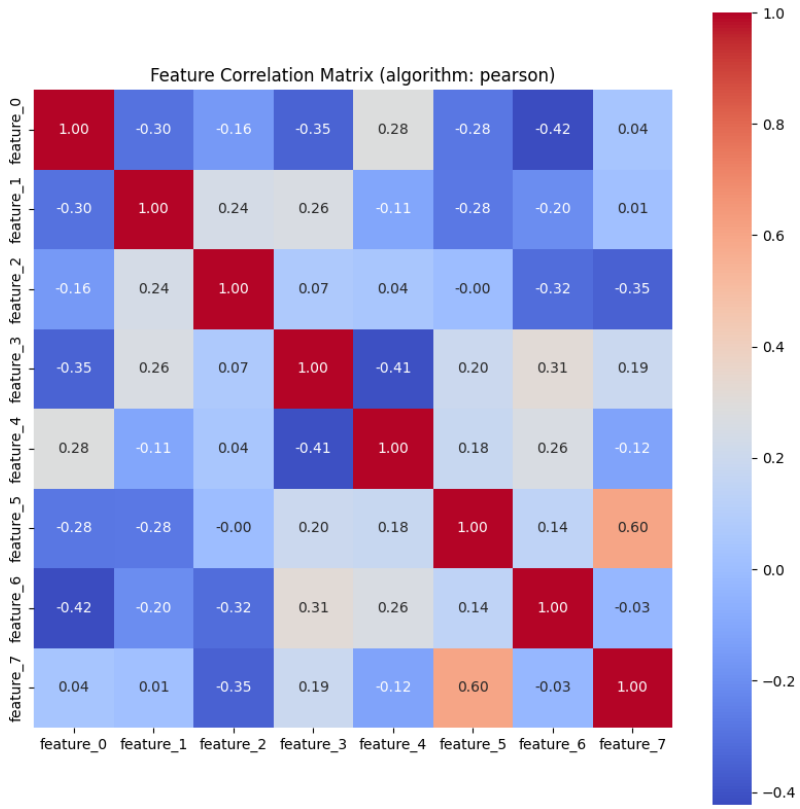


Table C.37: Word2vec 3 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Word2vec 3
Best Features	feature_4
	feature_3
	feature_1
	feature_2
	feature_0
	feature_6
	feature_7
	feature_5

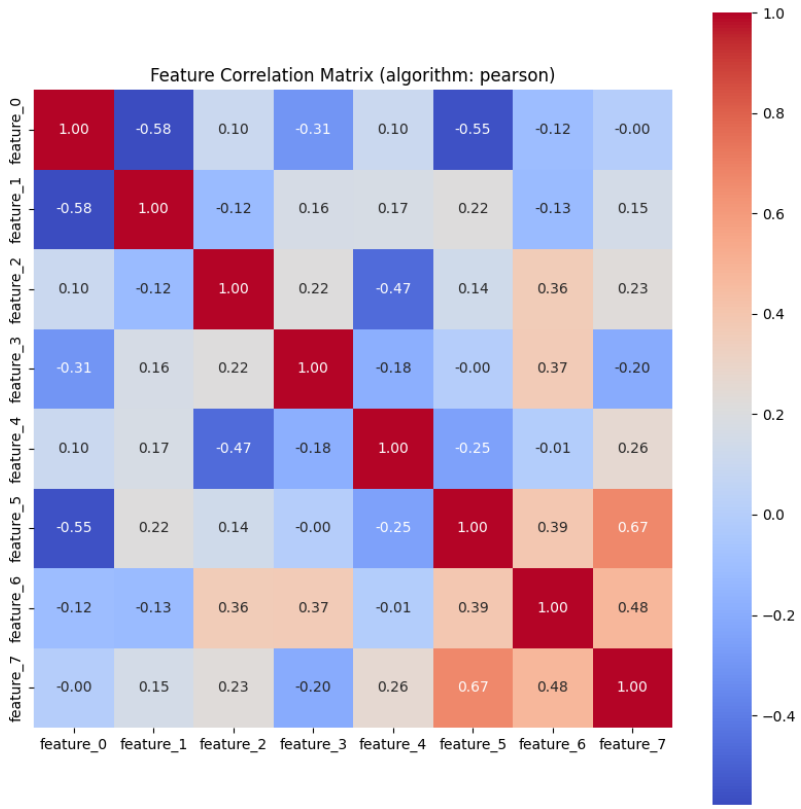


Table C.38: Word2vec 4 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Word2vec 4
Best Features	feature_8
	feature_11
	feature_9
	feature_10
	feature_0
	feature_1
	feature_13
	feature_2

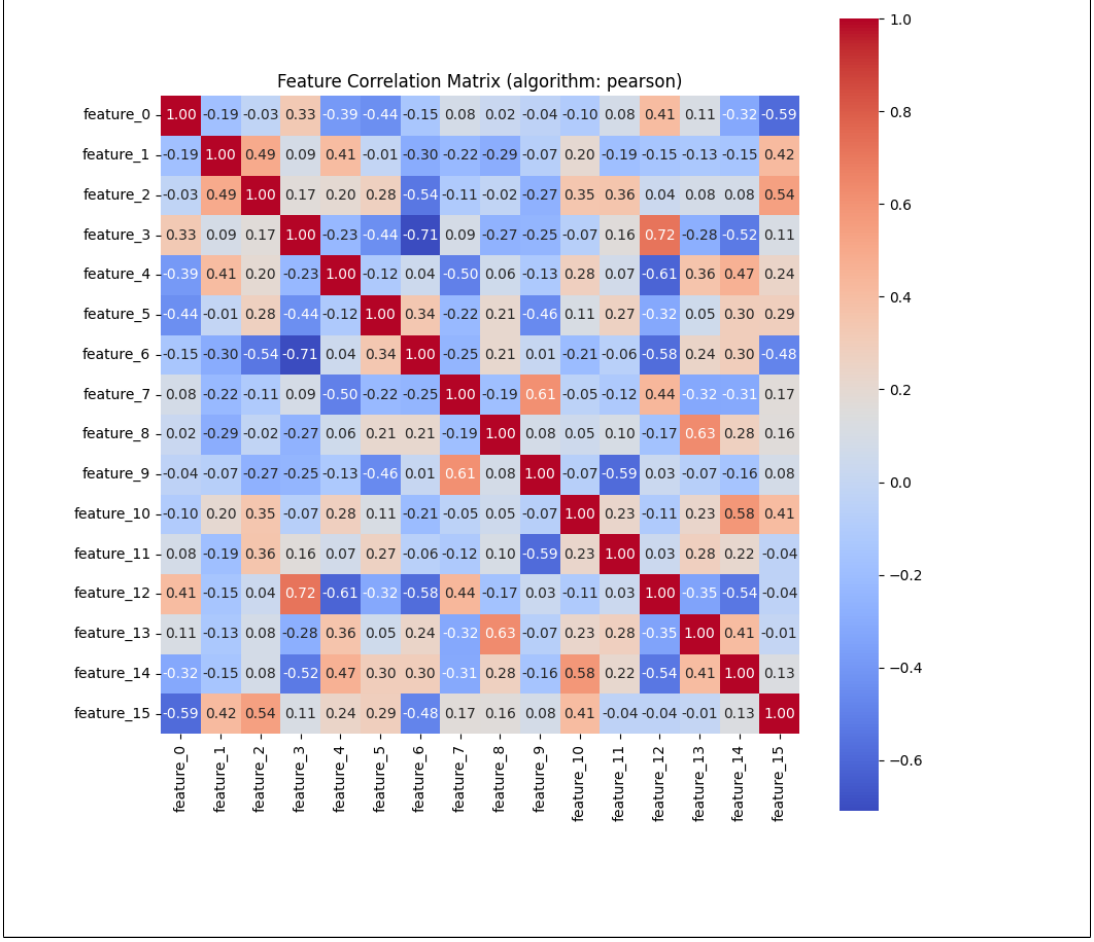


Table C.39: Word2vec 5 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Word2vec 5
Best Features	feature_12
	feature_8
	feature_3
	feature_0
	feature_7
	feature_2
	feature_14
	feature_4

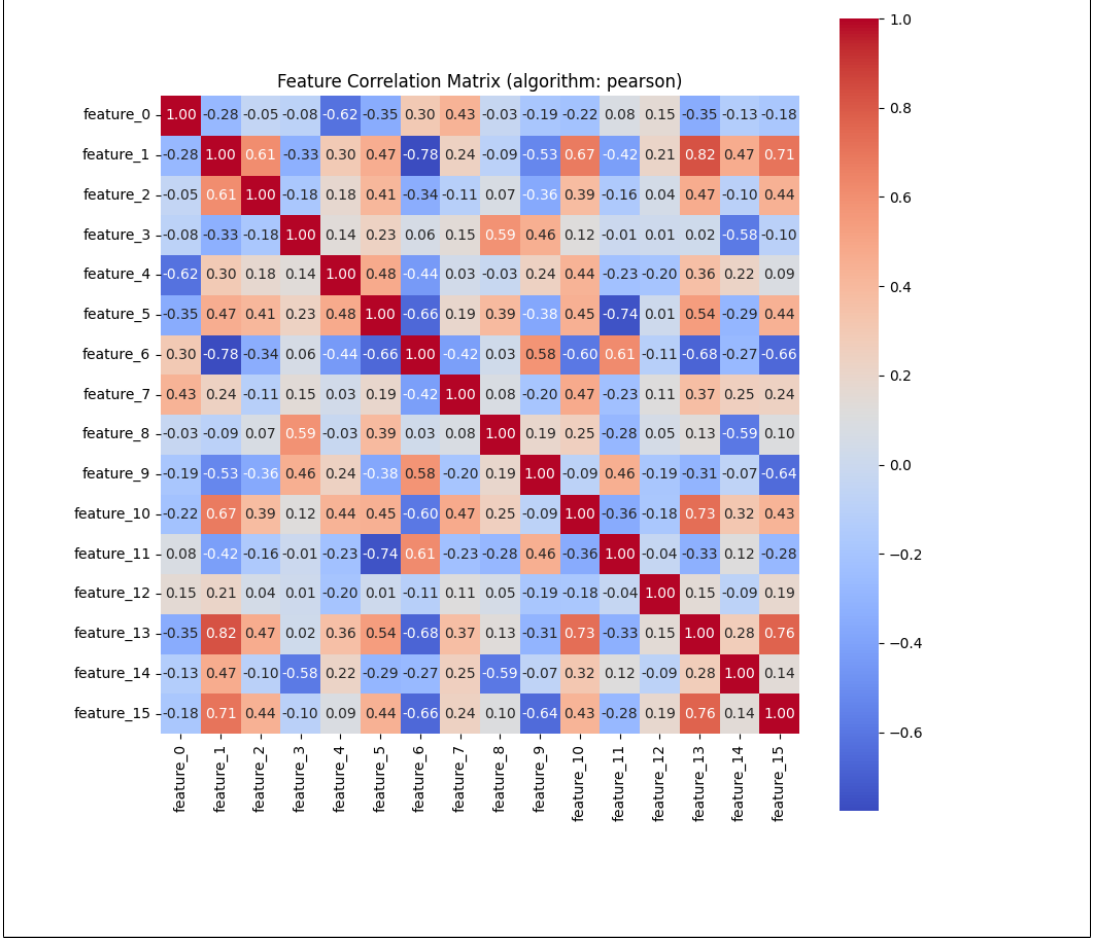


Table C.40: Word2vec 6 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Word2vec 6
Best Features	feature_11
	feature_3
	feature_12
	feature_14
	feature_13
	feature_8
	feature_5
	feature_6

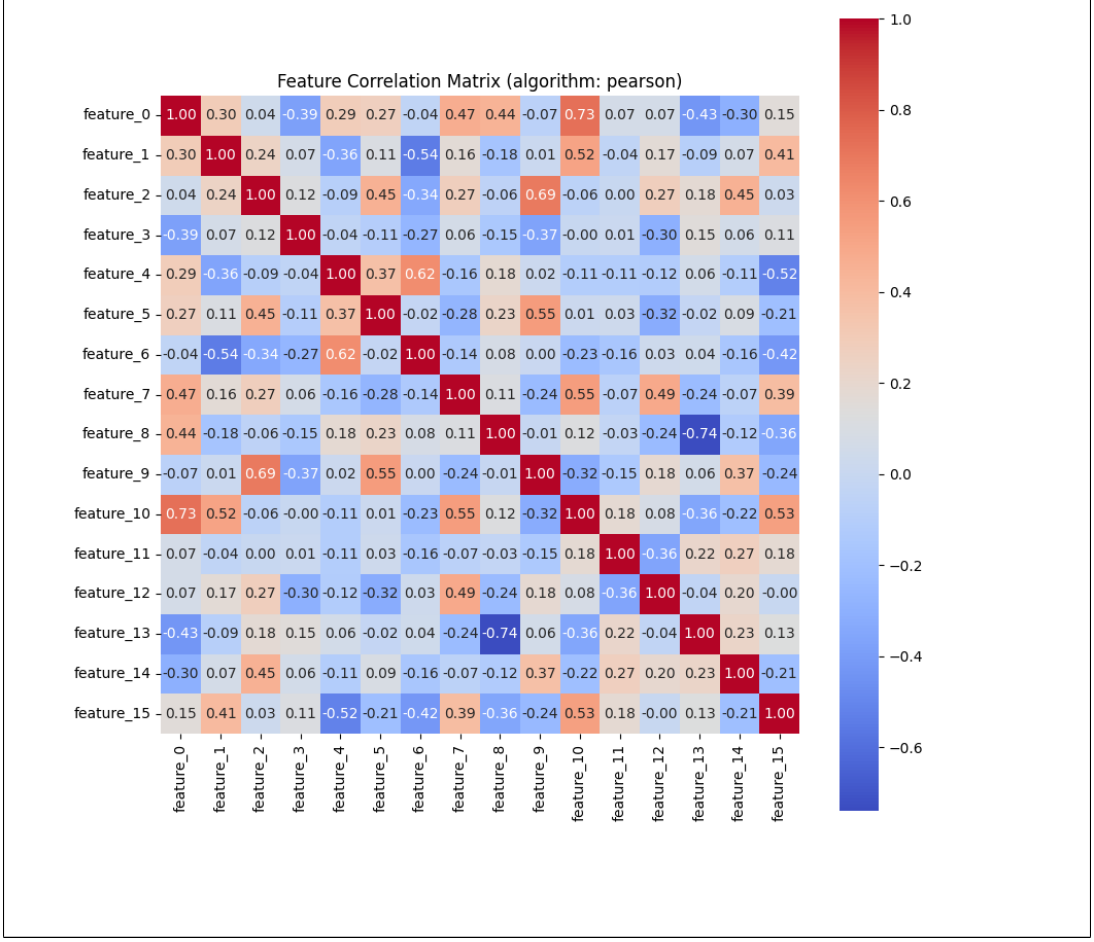


Table C.41: Word2vec 7 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Word2vec 7
Best Features	feature_5
	feature_12
	feature_2
	feature_3
	feature_6
	feature_14
	feature_13
	feature_11

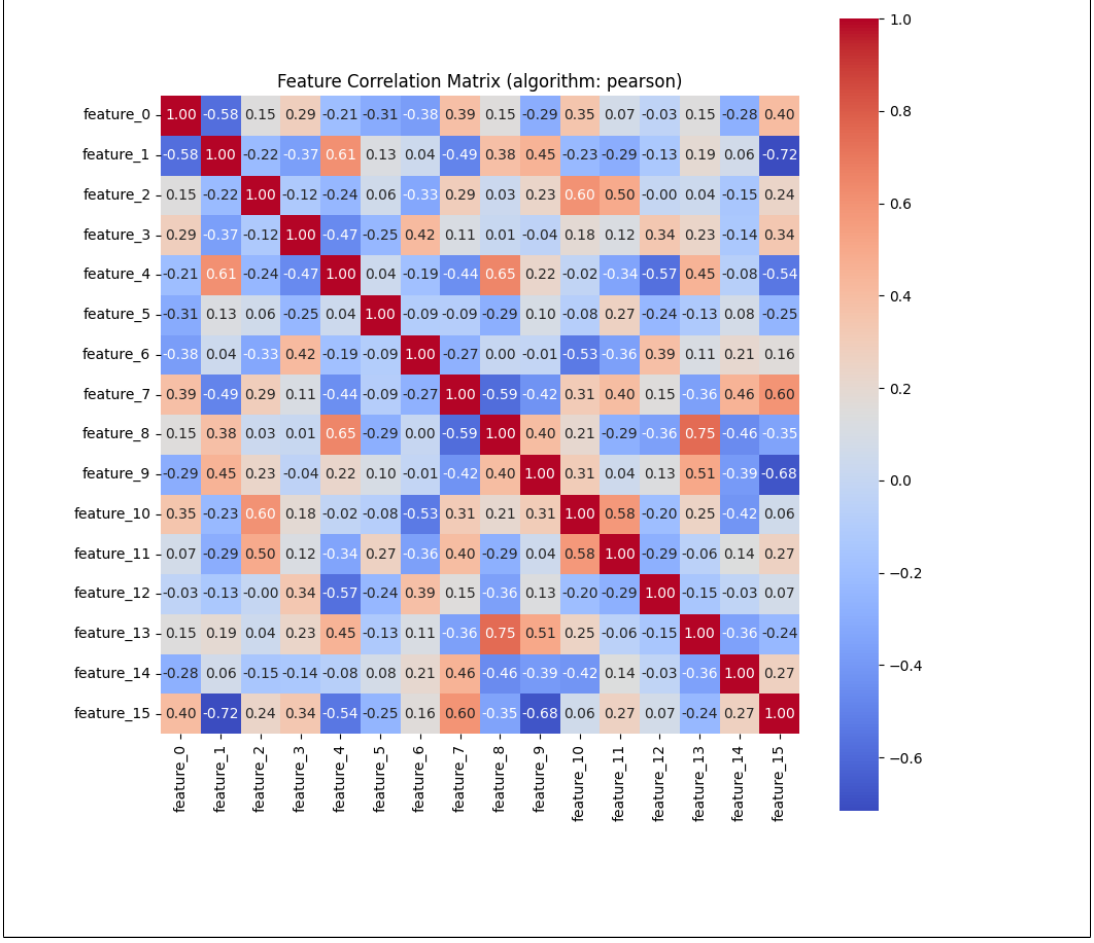


Table C.42: Word2vec 8 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Word2vec 8
Best Features	feature_33
	feature_61
	feature_86
	feature_93
	feature_70
	feature_32
	feature_10
	feature_42

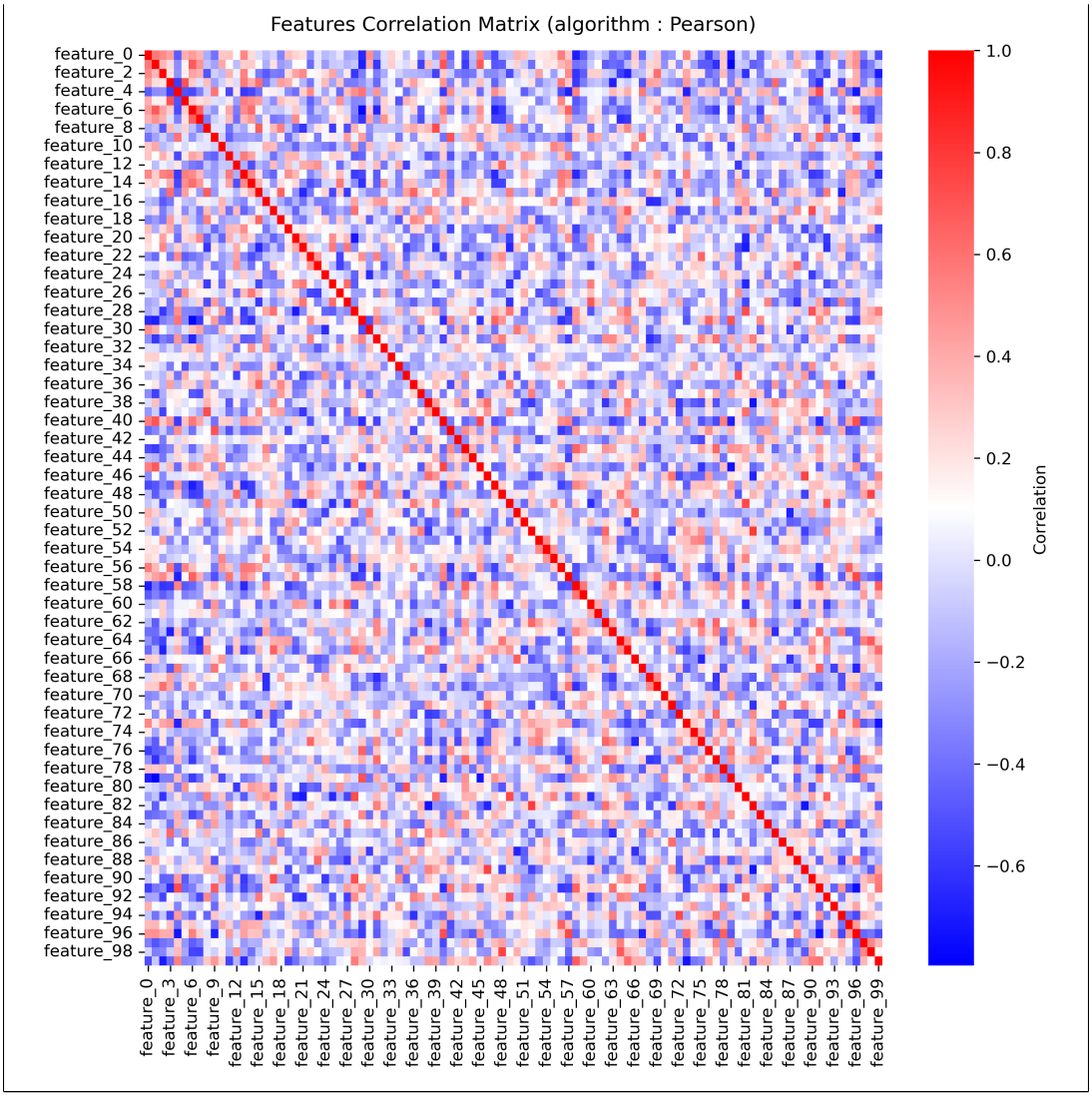


Table C.43: Word2vec 9 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Word2vec 9
Best Features	feature_15
	feature_16
	feature_69
	feature_57
	feature_30
	feature_75
	feature_28
	feature_42

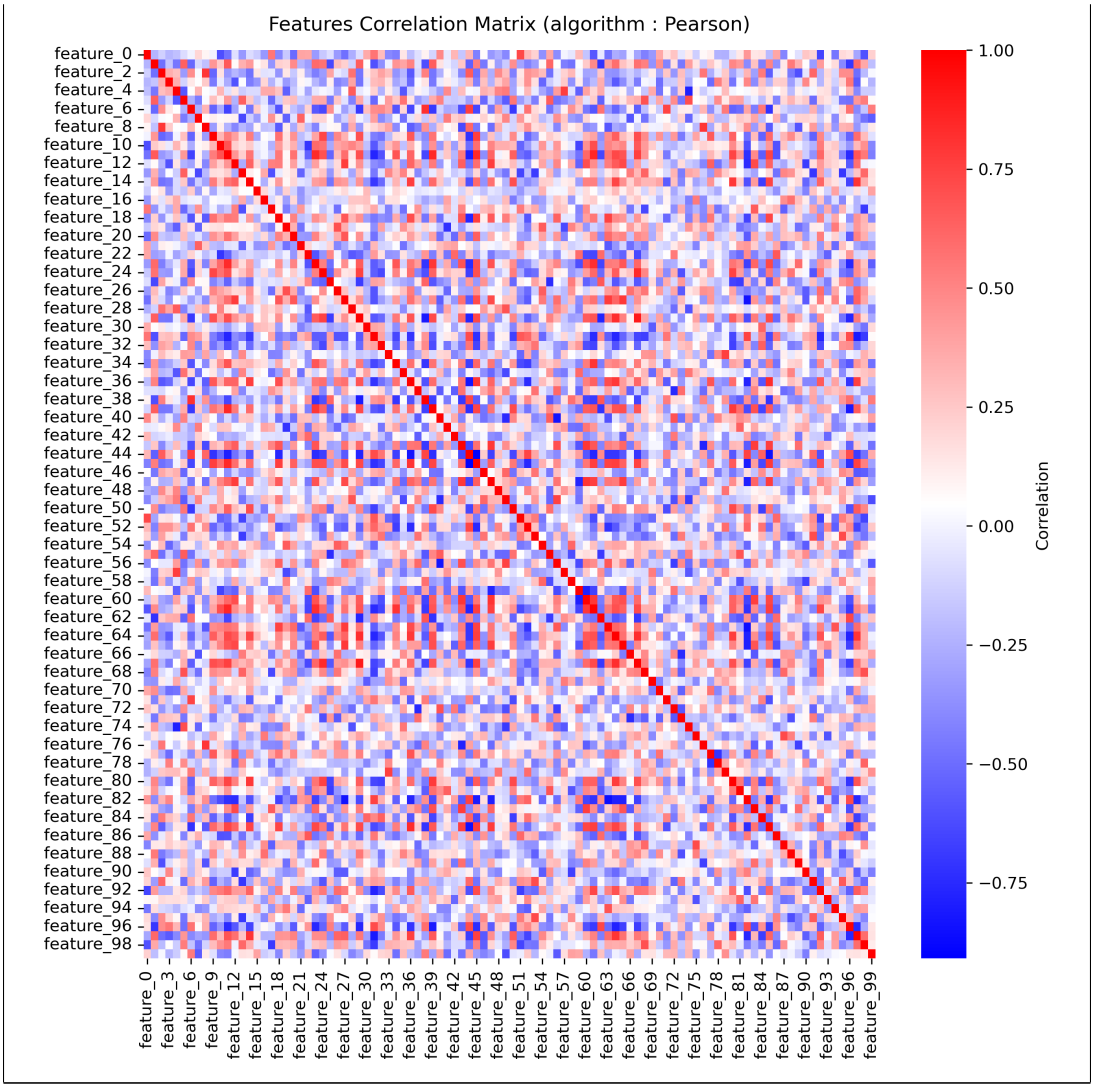


Table C.44: Word2vec 10 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Word2vec 10
Best Features	feature_78
	feature_25
	feature_10
	feature_86
	feature_68
	feature_92
	feature_71
	feature_12

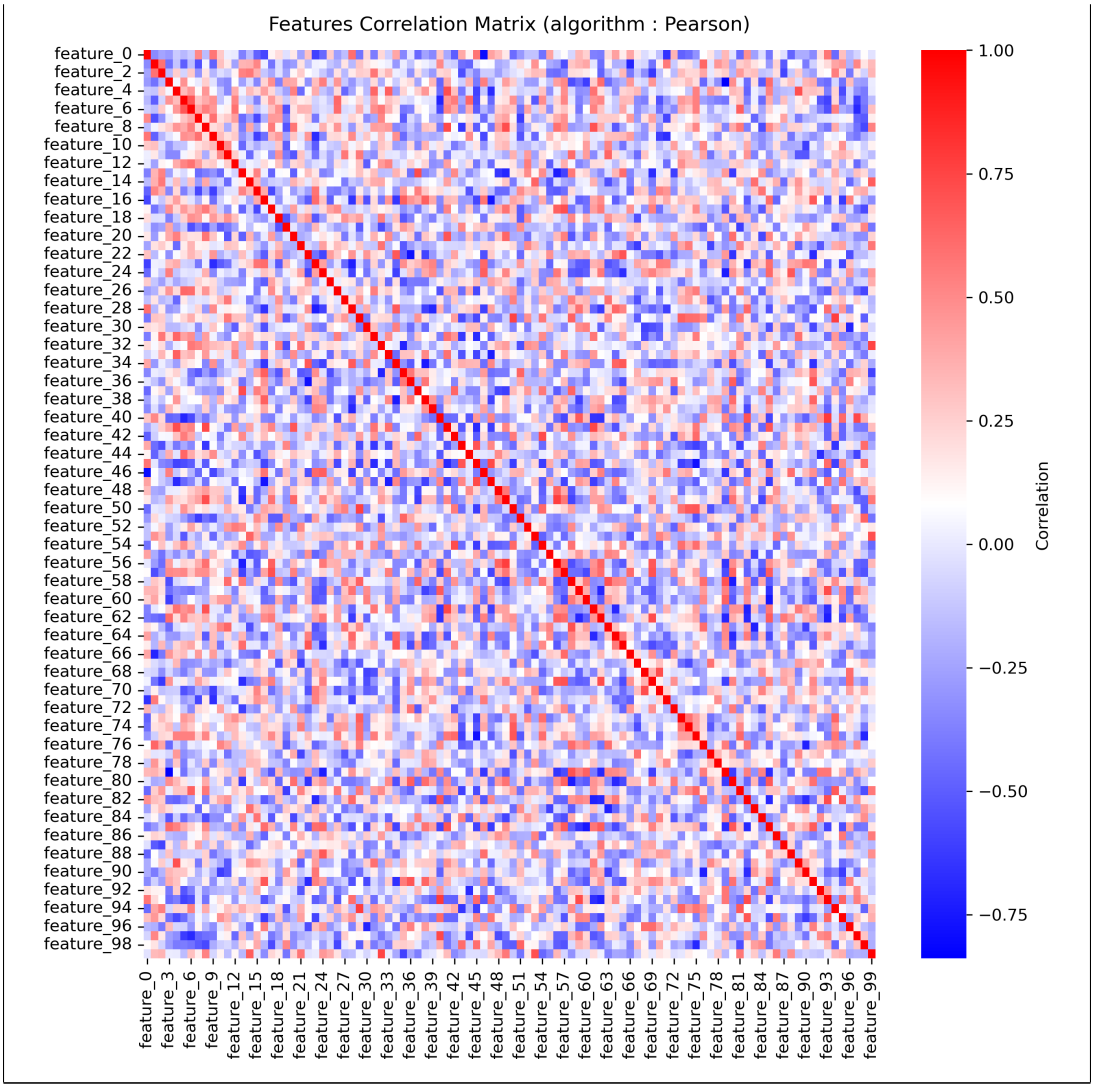
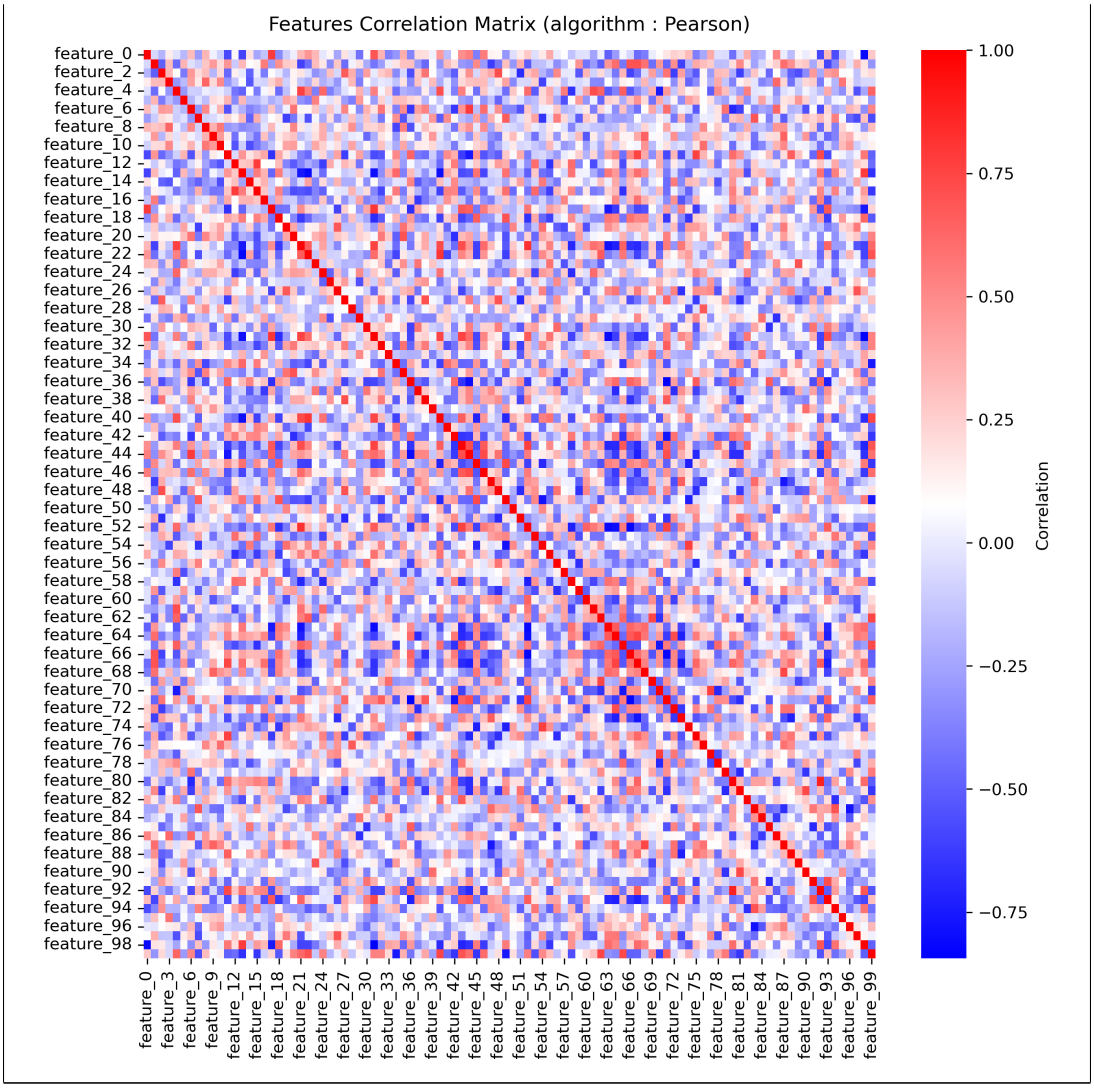


Table C.45: Word2vec 11 Feature Engineering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Dataset Name	27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate
Instance	Word2vec 11
Best Features	feature_28
	feature_39
	feature_84
	feature_76
	feature_89
	feature_77
	feature_50
	feature_95



C.4 Clustering results

C.4.1 25_filtered_chunk_extraction_-e_none_-s_activate

Table C.46: Word2vec 0 Clustering Results on 25_filtered_-chunk_extraction_-e_none_-s_activate

General Information				
Min Samples	937			
Total Duration	3327.69158 s			
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	657.12882 s
0.02	1	None	None	666.575554 s
0.03	1	None	None	674.458977 s
0.04	1	None	None	668.369706 s
0.05	1	None	None	661.124915 s

Label Association		
Cluster ID	Label	Number of Samples

Table C.47: Word2vec 1 Clustering Results on 25_filtered_-chunk_extraction_-e_none_-s_activate

General Information				
Min Samples		937		
Total Duration		3739.764487 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	774.223486 s
0.02	1	None	None	741.354391 s
0.03	1	None	None	731.504657 s
0.04	1	None	None	740.939556 s
0.05	1	None	None	751.715116 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.48: Word2vec 2 Clustering Results on 25_filtered_-chunk_extraction_-e_none_-s_activate

General Information				
Min Samples		937		
Total Duration		3869.848014 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	786.426559 s
0.02	1	None	None	803.378659 s
0.03	1	None	None	738.827762 s
0.04	1	None	None	779.099225 s
0.05	1	None	None	762.080643 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.49: Word2vec 3 Clustering Results on 25_filtered_-chunk_extraction_-e_none_-s_activate

General Information	
Min Samples	937
Total Duration	3634.492193 s
Clustering Information	

EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	2	0.21665062010288239	3676	747.692523 s
0.02	1	None	None	697.073419 s
0.03	1	None	None	694.980969 s
0.04	1	None	None	737.728848 s
0.05	1	None	None	756.257249 s
Best EPS Information				
0.01	2	0.21665062010288239	3676	747.692523 s
Label Association				
Cluster ID	Label		Number of Samples	
-1.0	0.0		3	
	1.0		2	
	2.0		1	
	4.0		2	
0.0	2.0		2	
	4.0		1	
1.0	0.0		2	

Table C.50: Word2vec 4 Clustering Results on 25_filtered_-chunk_extraction_-e_none_-s_activate

General Information				
Min Samples		937		
Total Duration		3811.38421 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	747.945469 s
0.02	1	None	None	750.852781 s
0.03	1	None	None	768.517218 s
0.04	1	None	None	761.567547 s
0.05	1	None	None	782.471201 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.51: Word2vec 5 Clustering Results on 25_filtered_-chunk_extraction_-e_none_-s_activate

General Information				
Min Samples		937		
Total Duration		3699.179144 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration

0.01	1	None	None	730.677446 s
0.02	1	None	None	748.488927 s
0.03	1	None	None	745.257242 s
0.04	1	None	None	749.60279 s
0.05	1	None	None	725.124423 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.52: Word2vec 6 Clustering Results on 25_filtered_-chunk_extraction_-e_none_-s_activate

General Information				
Min Samples		937		
Total Duration		3738.145444 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	741.121737 s
0.02	1	None	None	741.842508 s
0.03	1	None	None	772.812365 s
0.04	1	None	None	730.865507 s
0.05	1	None	None	751.47372 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.53: Word2vec 7 Clustering Results on 25_filtered_-chunk_extraction_-e_none_-s_activate

General Information				
Min Samples		937		
Total Duration		3656.017246 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	698.339024 s
0.02	1	None	None	717.647244 s
0.03	1	None	None	738.547307 s
0.04	1	None	None	761.364124 s
0.05	1	None	None	740.087451 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.54: Word2vec 8 Clustering Results on 25_filtered_-chunk_extraction_-e_none_-s_activate

General Information				
Min Samples		937		
Total Duration		4348.497164 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	842.152275 s
0.02	1	None	None	849.88952 s
0.03	1	None	None	946.85238 s
0.04	1	None	None	857.918362 s
0.05	1	None	None	851.607749 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.55: Word2vec 9 Clustering Results on 25_filtered_-chunk_extraction_-e_none_-s_activate

General Information				
Min Samples		937		
Total Duration		4377.139871 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	939.039679 s
0.02	1	None	None	761.224613 s
0.03	1	None	None	968.447559 s
0.04	1	None	None	879.374212 s
0.05	1	None	None	829.013344 s
Label Association				
Cluster ID	Label		Number of Samples	

C.4.2 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Table C.56: Transformers 0 Clustering Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

General Information				
Min Samples		937		
Total Duration		3588.869441 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	3	0.5749053359031677	1369	674.16573 s

0.02	3	0.5749053359031677	1369	664.292675 s
0.03	3	0.574790894985199	1371	699.778419 s
0.04	3	0.574790894985199	1371	770.27842 s
0.05	3	0.574790894985199	1371	776.637103 s
Best EPS Information				
0.01	3	0.5749053359031677	1369	674.16573 s
Label Association				
Cluster ID	Label		Number of Samples	
-1.0	0.0		31	
	1.0		40	
	2.0		56	
	4.0		33	
0.0	0.0		35	
	1.0		40	
	2.0		26	
	4.0		32	
1.0	0.0		77	
	1.0		91	
	2.0		76	
	4.0		96	
2.0	0.0		32	
	1.0		45	
	2.0		45	
	4.0		42	

Table C.57: Transformers 1 Clustering Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

General Information				
Min Samples		937		
Total Duration		3739.406415 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	3	0.7379494309425354	1299	760.134469 s
0.02	3	0.7379494309425354	1299	731.259721 s
0.03	3	0.7379494309425354	1299	738.923279 s
0.04	3	0.7379494309425354	1299	761.430407 s
0.05	3	0.3740614056587219	2696	743.313265 s
Best EPS Information				
0.01	3	0.7379494309425354	1299	760.134469 s
Label Association				
Cluster ID	Label		Number of Samples	
-1.0	0.0		29	

	1.0	37
	2.0	56
	4.0	32
0.0	0.0	37
	1.0	43
	2.0	26
	4.0	33
1.0	0.0	77
	1.0	91
	2.0	76
	4.0	96
2.0	0.0	32
	1.0	45
	2.0	45
	4.0	42

Table C.58: Word2vec 0 Clustering Results on 26_filtered_-chunk_extraction_-e_only-max-entropy_-s_none

General Information				
Min Samples		937		
Total Duration		5731.762181 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	1130.121242 s
0.02	1	None	None	1166.507598 s
0.03	1	None	None	1200.210931 s
0.04	1	None	None	1145.306208 s
0.05	1	None	None	1089.581316 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.59: Word2vec 1 Clustering Results on 26_filtered_-chunk_extraction_-e_only-max-entropy_-s_none

General Information				
Min Samples		937		
Total Duration		4650.233523 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	1000.668362 s
0.02	1	None	None	935.551617 s
0.03	1	None	None	891.011774 s

0.04	1	None	None	915.492114 s
0.05	1	None	None	907.477479 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.60: Word2vec 2 Clustering Results on 26_filtered_-chunk_extraction_-e_only-max-entropy_-s_none

General Information				
Min Samples	937			
Total Duration	4900.190038 s			
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	988.473985 s
0.02	1	None	None	961.787699 s
0.03	1	None	None	992.206004 s
0.04	1	None	None	1030.09568 s
0.05	1	None	None	927.586959 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.61: Word2vec 3 Clustering Results on 26_filtered_-chunk_extraction_-e_only-max-entropy_-s_none

General Information				
Min Samples	937			
Total Duration	4225.839363 s			
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	810.379002 s
0.02	1	None	None	820.338574 s
0.03	1	None	None	867.738331 s
0.04	1	None	None	889.716803 s
0.05	1	None	None	837.634967 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.62: Word2vec 4 Clustering Results on 26_filtered_-chunk_extraction_-e_only-max-entropy_-s_none

General Information	
Min Samples	937

Total Duration		4176.399826 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	891.282476 s
0.02	1	None	None	896.741625 s
0.03	1	None	None	786.737323 s
0.04	1	None	None	788.139638 s
0.05	1	None	None	813.46316 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.63: Word2vec 5 Clustering Results on 26_filtered_-chunk_extraction_-e_only-max-entropy_-s_none

General Information				
Min Samples	937			
Total Duration	4382.460602 s			
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	849.683867 s
0.02	1	None	None	887.746338 s
0.03	1	None	None	858.117839 s
0.04	1	None	None	885.070443 s
0.05	1	None	None	901.803542 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.64: Word2vec 6 Clustering Results on 26_filtered_-chunk_extraction_-e_only-max-entropy_-s_none

General Information				
Min Samples	937			
Total Duration	5024.658551 s			
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	812.30807 s
0.02	1	None	None	869.36774 s
0.03	1	None	None	1322.429996 s
0.04	1	None	None	1007.891293 s
0.05	1	None	None	1012.629016 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.65: Word2vec 7 Clustering Results on 26_filtered_-chunk_extraction_-e_only-max-entropy_-s_none

General Information				
Min Samples		937		
Total Duration		4427.70411 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	894.277567 s
0.02	1	None	None	862.553169 s
0.03	1	None	None	879.521478 s
0.04	1	None	None	916.011661 s
0.05	1	None	None	875.30631 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.66: Word2vec 8 Clustering Results on 26_filtered_-chunk_extraction_-e_only-max-entropy_-s_none

General Information				
Min Samples		937		
Total Duration		4808.459968 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	953.453177 s
0.02	1	None	None	933.502783 s
0.03	1	None	None	1026.357889 s
0.04	1	None	None	969.195393 s
0.05	1	None	None	925.920198 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.67: Word2vec 9 Clustering Results on 26_filtered_-chunk_extraction_-e_only-max-entropy_-s_none

General Information				
Min Samples		937		
Total Duration		4254.490446 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	878.185887 s
0.02	1	None	None	835.512919 s
0.03	1	None	None	829.098421 s

0.04	1	None	None	843.143088 s
0.05	1	None	None	868.51433 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.68: Word2vec 10 Clustering Results on 26_filtered_-chunk_extraction_-e_only-max-entropy_-s_none

General Information				
Min Samples	937			
Total Duration	4651.313538 s			
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	791.83869 s
0.02	1	None	None	940.217059 s
0.03	1	None	None	984.917923 s
0.04	1	None	None	1008.4868 s
0.05	1	None	None	925.815858 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.69: Word2vec 11 Clustering Results on 26_filtered_-chunk_extraction_-e_only-max-entropy_-s_none

General Information				
Min Samples	937			
Total Duration	4173.674751 s			
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	841.864691 s
0.02	1	None	None	814.169629 s
0.03	1	None	None	862.086083 s
0.04	1	None	None	843.033758 s
0.05	1	None	None	812.482109 s
Label Association				
Cluster ID	Label		Number of Samples	

C.4.3 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Table C.70: Transformers 0 Clustering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples		937		
Total Duration		3796.324605 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	3	0.2465989887714386	2274	709.617939 s
0.02	3	0.24514129757881165	2281	753.217155 s
0.03	2	0.1906411200761795	3242	820.552446 s
0.04	2	0.1906411200761795	3242	756.897654 s
0.05	2	0.1906411200761795	3242	751.836631 s
Best EPS Information				
0.01	3	0.2465989887714386	2274	709.617939 s
Label Association				
Cluster ID	Label		Number of Samples	
-1.0	0.0		91	
	1.0		95	
	2.0		99	
	4.0		97	
0.0	0.0		128	
	1.0		129	
	2.0		94	
	4.0		110	
1.0	0.0		50	
	1.0		40	
	2.0		44	
	4.0		33	
2.0	0.0		52	
	1.0		51	
	2.0		53	
	4.0		50	

Table C.71: Transformers 1 Clustering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples		937		
Total Duration		3943.383724 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	4	0.5144023299217224	981	810.887225 s
0.02	3	0.8070926070213318	533	798.792131 s
0.03	3	0.8070926070213318	533	772.422919 s
0.04	3	0.8070926070213318	533	775.663572 s

0.05	3	0.8070926070213318	533	781.48437 s
Best EPS Information				
0.02	3	0.8070926070213318	533	798.792131 s
Label Association				
Cluster ID	Label		Number of Samples	
-1.0	0.0		19	
	1.0		20	
	2.0		20	
	4.0		20	
0.0	0.0		182	
	1.0		155	
	2.0		138	
	4.0		145	
1.0	0.0		74	
	1.0		96	
	2.0		81	
	4.0		85	
2.0	0.0		46	
	1.0		44	
	2.0		51	
	4.0		40	

Table C.72: Transformers 2 Clustering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples		937		
Total Duration		3895.618311 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	2	0.907838761806488	0	790.283896 s
0.02	2	0.907838761806488	0	802.295121 s
0.03	2	0.907838761806488	0	787.142639 s
0.04	2	0.907838761806488	0	725.182204 s
0.05	2	0.907838761806488	0	785.995246 s
Best EPS Information				
0.01	2	0.907838761806488	0	790.283896 s
Label Association				
Cluster ID	Label		Number of Samples	
0.0	0.0		264	
	1.0		256	
	2.0		228	
	4.0		236	

1.0	0.0	57
	1.0	59
	2.0	62
	4.0	54

Table C.73: Transformers 3 Clustering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples		937		
Total Duration		3939.333846 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	3	0.36908280849456787	2637	806.707338 s
0.02	3	0.36908280849456787	2637	782.581847 s
0.03	3	0.36908280849456787	2637	775.659231 s
0.04	3	0.36908280849456787	2637	783.727976 s
0.05	3	0.36908280849456787	2637	786.255646 s
Best EPS Information				
0.01	3	0.36908280849456787	2637	806.707338 s
Label Association				
Cluster ID	Label		Number of Samples	
-1.0	0.0		134	
	1.0		97	
	2.0		114	
	4.0		111	
0.0	0.0		81	
	1.0		86	
	2.0		53	
	4.0		72	
1.0	0.0		49	
	1.0		73	
	2.0		61	
	4.0		53	
2.0	0.0		57	
	1.0		59	
	2.0		62	
	4.0		54	

Table C.74: Transformers 4 Clustering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples		937		
Total Duration		3803.240679 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	3	0.6185895800590515	1103	794.276663 s
0.02	3	0.6185895800590515	1103	746.079328 s
0.03	3	0.6186758875846863	1105	737.198024 s
0.04	2	0.47322237491607666	2390	732.879635 s
0.05	2	0.47322237491607666	2390	788.087243 s
Best EPS Information				
0.03	3	0.6186758875846863	1105	737.198024 s
Label Association				
Cluster ID	Label		Number of Samples	
-1.0	0.0		44	
	1.0		43	
	2.0		43	
	4.0		53	
0.0	0.0		184	
	1.0		159	
	2.0		140	
	4.0		146	
1.0	0.0		47	
	1.0		69	
	2.0		59	
	4.0		52	
2.0	0.0		46	
	1.0		44	
	2.0		48	
	4.0		39	

Table C.75: Transformers 5 Clustering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples		937		
Total Duration		3935.993827 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	764.413697 s
0.02	1	None	None	783.772212 s
0.03	1	None	None	811.087261 s
0.04	1	None	None	772.993991 s

0.05	1	None	None	803.689671 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.76: Transformers 6 Clustering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples		937		
Total Duration		3973.074733 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	3	0.37613049149513245	2636	806.924699 s
0.02	3	0.37613049149513245	2636	759.43715 s
0.03	3	0.37613049149513245	2636	762.661041 s
0.04	3	0.37613049149513245	2636	821.687468 s
0.05	3	0.37613049149513245	2636	818.360119 s
Best EPS Information				
0.01	3	0.37613049149513245	2636	806.924699 s
Label Association				
Cluster ID	Label		Number of Samples	
-1.0	0.0		134	
	1.0		97	
	2.0		114	
	4.0		111	
0.0	0.0		81	
	1.0		86	
	2.0		53	
	4.0		72	
1.0	0.0		49	
	1.0		73	
	2.0		61	
	4.0		53	
2.0	0.0		57	
	1.0		59	
	2.0		62	
	4.0		54	

Table C.77: Transformers 7 Clustering Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

General Information	
Min Samples	937

Total Duration		3806.555889 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	768.61293 s
0.02	1	None	None	748.107116 s
0.03	1	None	None	727.759616 s
0.04	1	None	None	757.47944 s
0.05	1	None	None	804.568675 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.78: Word2vec 0 Clustering Results on 27_filtered_-chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples	937			
Total Duration	4160.518566 s			
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	780.8798 s
0.02	1	None	None	839.874733 s
0.03	1	None	None	857.032421 s
0.04	1	None	None	802.275627 s
0.05	1	None	None	880.429649 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.79: Word2vec 1 Clustering Results on 27_filtered_-chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples	937			
Total Duration	4102.458592 s			
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	2	0.2816171944141388	3355	861.723101 s
0.02	2	0.24786975979804993	3822	806.529987 s
0.03	2	0.23544950783252716	3938	786.201026 s
0.04	2	0.2358776181936264	3952	838.073615 s
0.05	2	0.2358776181936264	3952	805.644238 s
Best EPS Information				
0.01	2	0.2816171944141388	3355	861.723101 s
Label Association				

Cluster ID	Label	Number of Samples
-1.0	0.0	149
	1.0	140
	2.0	126
	4.0	135
0.0	0.0	60
	1.0	66
	2.0	63
	4.0	72
1.0	0.0	112
	1.0	109
	2.0	101
	4.0	83

Table C.80: Word2vec 2 Clustering Results on 27_filtered_-chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples		937		
Total Duration		4253.37304 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	878.746583 s
0.02	1	None	None	865.696646 s
0.03	1	None	None	812.45957 s
0.04	1	None	None	868.793452 s
0.05	1	None	None	827.650692 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.81: Word2vec 3 Clustering Results on 27_filtered_-chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples		937		
Total Duration		3907.871639 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	746.159505 s
0.02	1	None	None	821.070716 s
0.03	1	None	None	812.788339 s
0.04	1	None	None	788.387325 s

0.05	1	None	None	739.434204 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.82: Word2vec 4 Clustering Results on 27_filtered_-chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples		937		
Total Duration		3948.822965 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	817.582107 s
0.02	1	None	None	794.100492 s
0.03	1	None	None	901.912435 s
0.04	1	None	None	724.102389 s
0.05	1	None	None	711.091397 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.83: Word2vec 5 Clustering Results on 27_filtered_-chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples		937		
Total Duration		3847.688397 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	686.551731 s
0.02	1	None	None	798.569698 s
0.03	1	None	None	742.917964 s
0.04	1	None	None	817.425584 s
0.05	1	None	None	802.191171 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.84: Word2vec 6 Clustering Results on 27_filtered_-chunk_extraction_-e_only-max-entropy_-s_activate

General Information	
Min Samples	937
Total Duration	3793.274982 s

Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	805.974383 s
0.02	1	None	None	782.97422 s
0.03	1	None	None	747.737812 s
0.04	1	None	None	723.384303 s
0.05	1	None	None	733.170547 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.85: Word2vec 7 Clustering Results on 27_filtered_-chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples	937			
Total Duration	3940.443007 s			
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	773.901169 s
0.02	1	None	None	821.871725 s
0.03	1	None	None	824.555681 s
0.04	1	None	None	786.808999 s
0.05	1	None	None	733.269815 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.86: Word2vec 8 Clustering Results on 27_filtered_-chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples	937			
Total Duration	3934.021547 s			
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	864.845278 s
0.02	1	None	None	723.613036 s
0.03	1	None	None	840.656249 s
0.04	1	None	None	716.829823 s
0.05	1	None	None	788.044763 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.87: Word2vec 9 Clustering Results on 27_filtered_-chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples		937		
Total Duration		3803.230109 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	792.123074 s
0.02	1	None	None	808.30978 s
0.03	1	None	None	711.558043 s
0.04	1	None	None	752.351878 s
0.05	1	None	None	738.852176 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.88: Word2vec 10 Clustering Results on 27_filtered_-chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples		937		
Total Duration		4053.433277 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	817.598159 s
0.02	1	None	None	842.449286 s
0.03	1	None	None	793.654269 s
0.04	1	None	None	780.799898 s
0.05	1	None	None	818.895812 s
Label Association				
Cluster ID	Label		Number of Samples	

Table C.89: Word2vec 11 Clustering Results on 27_filtered_-chunk_extraction_-e_only-max-entropy_-s_activate

General Information				
Min Samples		937		
Total Duration		3697.510704 s		
Clustering Information				
EPS	Number of Clusters	Silhouette Score	Noise Points	Duration
0.01	1	None	None	786.951855 s
0.02	1	None	None	786.121058 s
0.03	1	None	None	673.828587 s

0.04	1	None	None	699.455122 s
0.05	1	None	None	751.119681 s
Label Association				
Cluster ID	Label		Number of Samples	

C.5 Classification results

C.5.1 25_filtered_chunk_extraction_-e_none_-s_activate

Table C.90: Word2vec 0 Classification Results on 25_filtered_chunk_extraction_-e_none_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9997229995364942
	Recall	0.898748486144646
	F1 Score	0.9465504635430876
	Support	4437346.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	24506437
1.0	Precision	0.12456421698507879
	Recall	0.9566265060240964
	F1 Score	0.22042629322310991
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.01213194965898356
	Recall	0.9386880856760375
	F1 Score	0.02395430507918722
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.21909251620506776
	Recall	0.9954484605087015
	F1 Score	0.3591403042743298
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.3388779205964061
	Recall	0.9473778845883704
	F1 Score	0.38751784152992863
	Support	4467226.0
	Final Samples (after rebalancing)	83856

	Initial Samples (before rebalancing)	24674149
Weighted Avg	Precision	0.9938543428480903
	Recall	0.8991530762043382
	F1 Score	0.94164533533744
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
	Accuracy	0.899153076204338
	True Positives	21438
	True Negatives	3988058
	False Positives	150642
	False Negatives	940
	AUC	0.89
	Duration (seconds)	16.166093

Table C.91: Word2vec 1 Classification Results on 25_filtered_chunk_extraction_-e_none_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9999557769913873
	Recall	0.9834822436654703
	F1 Score	0.9916505994939552
	Support	4437346.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	24506437
1.0	Precision	0.40826033460048117
	Recall	0.992012494422133
	F1 Score	0.5784577755226832
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.10221483942414175
	Recall	0.9884872824631861
	F1 Score	0.1852716095847447
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.2999435893303248
	Recall	0.9965194109772423
	F1 Score	0.4611000991080278
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964

Macro Avg	Precision	0.4525936350865838
	Recall	0.9901253578820078
	F1 Score	0.5541200209273527
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
Weighted Avg	Precision	0.9956516511677312
	Recall	0.9835401208714312
	F1 Score	0.9884600103383028
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
	Accuracy	0.9835401208714312
	True Positives	22231
	True Negatives	4364051
	False Positives	32210
	False Negatives	149
	AUC	0.98
	Duration (seconds)	15.441376

Table C.92: Word2vec 2 Classification Results on 25_filtered_chunk_extraction_-e_none_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9993930094030391
	Recall	0.847475495487618
	F1 Score	0.9171861273153783
	Support	4437346.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	24506437
1.0	Precision	0.054674257899082965
	Recall	0.8981704596162428
	F1 Score	0.10307409474746129
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.010676279288457017
	Recall	0.9119143239625167
	F1 Score	0.02110546536125914
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.21588996575938715

	Recall	0.9959839357429718
	F1 Score	0.3548602499284556
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.32015837808749154
	Recall	0.9133860537023373
	F1 Score	0.34905648433813863
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
Weighted Avg	Precision	0.9931720597407229
	Recall	0.8479078515391879
	F1 Score	0.9118827468563897
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
	Accuracy	0.8479078515391879
	True Positives	20128
	True Negatives	3760542
	False Positives	347916
	False Negatives	2120
	AUC	0.84
	Duration (seconds)	16.571076

Table C.93: Word2vec 3 Classification Results on 25_filtered_chunk_extraction_-e_none_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9999617471431186
	Recall	0.9838132974079551
	F1 Score	0.9918217959650952
	Support	4437346.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	24506437
1.0	Precision	0.35848600570406536
	Recall	0.9927710843373494
	F1 Score	0.5267606634229499
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.19214470284237725
	Recall	0.9954484605087015

	F1 Score	0.32211392679228934
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.18481717011128776
	Recall	0.9959839357429718
	F1 Score	0.31177974269790054
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.4338524064502122
	Recall	0.9920041944992445
	F1 Score	0.5381190322195588
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
Weighted Avg	Precision	0.9953868201030884
	Recall	0.9838781382450765
	F1 Score	0.9883602885462669
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
	Accuracy	0.9838781382450765
	True Positives	22248
	True Negatives	4365520
	False Positives	39808
	False Negatives	140
	AUC	0.98
	Duration (seconds)	15.149247

Table C.94: Word2vec 4 Classification Results on 25_filtered_chunk_extraction_-e_none_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9993063812635229
	Recall	0.7805289468073934
	F1 Score	0.8764715982471464
	Support	4437346.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	24506437
1.0	Precision	0.04910597619681468
	Recall	0.8971887550200803
	F1 Score	0.0931154495416243

	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.005415937328616509
	Recall	0.8275769745649264
	F1 Score	0.010761448182460685
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.17539304093291158
	Recall	0.9946452476572959
	F1 Score	0.2982019585808316
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.30730533393046644
	Recall	0.874984981012424
	F1 Score	0.31963761363801574
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
Weighted Avg	Precision	0.9930198203839576
	Recall	0.7813325316426792
	F1 Score	0.8713345678378641
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
	Accuracy	0.7813325316426794
	True Positives	20106
	True Negatives	3463477
	False Positives	389223
	False Negatives	1991
	AUC	0.79
	Duration (seconds)	16.491675

Table C.95: Word2vec 5 Classification Results on 25_filtered_chunk_extraction_-e_none_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9998968151108742
	Recall	0.97834899509752
	F1 Score	0.9890055515010877
	Support	4437346.0

	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	24506437
1.0	Precision	0.2731295103012476
	Recall	0.9778670236501562
	F1 Score	0.426994534454371
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.11532321177023008
	Recall	0.9863453815261044
	F1 Score	0.2065022421524664
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.27913258797929014
	Recall	0.9959839357429718
	F1 Score	0.43605673426327507
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.4168705312904105
	Recall	0.9846363340041882
	F1 Score	0.5146397655928001
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
Weighted Avg	Precision	0.9949087549371564
	Recall	0.9783680073495274
	F1 Score	0.9850696457320897
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
	Accuracy	0.9783680073495274
	True Positives	21914
	True Negatives	4341273
	False Positives	58308
	False Negatives	394
	AUC	0.97
	Duration (seconds)	17.036017

Table C.96: Word2vec 6 Classification Results on 25_filtered_chunk_extraction_-e_none_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9992139366188213
	Recall	0.8061237505481881
	F1 Score	0.8923428474494246
	Support	4437346.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	24506437
1.0	Precision	0.07611314742286651
	Recall	0.8654618473895582
	F1 Score	0.1399209315076399
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.004903192742289175
	Recall	0.7991967871485943
	F1 Score	0.009746588693957116
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.1554068274613798
	Recall	0.9884872824631861
	F1 Score	0.2685872253746544
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.30890927606133917
	Recall	0.8648174168873817
	F1 Score	0.32764939825641903
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
Weighted Avg	Precision	0.9930463375613833
	Recall	0.8065681028898023
	F1 Score	0.8873088510921424
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
	Accuracy	0.8065681028898023
	True Positives	19395
	True Negatives	3577050
	False Positives	235256
	False Negatives	2353
	AUC	0.79

	Duration (seconds)	15.789276

Table C.97: Word2vec 7 Classification Results on 25_filtered_chunk_extraction_-e_none_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9998840303295341
	Recall	0.9754064704442701
	F1 Score	0.9874935889128804
	Support	4437346.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	24506437
1.0	Precision	0.30222534357872705
	Recall	0.9793395805443998
	F1 Score	0.4619059645578146
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.08182141586623977
	Recall	0.9852744310575636
	F1 Score	0.15109523516248896
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.17796691211628574
	Recall	0.9965194109772423
	F1 Score	0.30200008113919424
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.3904744254726967
	Recall	0.984134973255869
	F1 Score	0.47562371744309456
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
Weighted Avg	Precision	0.9949294219278925
	Recall	0.9754521038335647
	F1 Score	0.9835845221308127
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
	Accuracy	0.9754521038335647

	True Positives	21947
	True Negatives	4328216
	False Positives	50666
	False Negatives	441
	AUC	0.97
	Duration (seconds)	15.530361

Table C.98: Word2vec 8 Classification Results on 25_filtered_chunk_extraction_-e_none_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9982576655833362
	Recall	0.6211882057428021
	F1 Score	0.7658247788731783
	Support	4437346.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	24506437
1.0	Precision	0.01701693459627122
	Recall	0.7285586791610889
	F1 Score	0.033257083960540446
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.0037639299202153875
	Recall	0.7156626506024096
	F1 Score	0.0074884752282191975
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.10178718724223261
	Recall	0.9911646586345382
	F1 Score	0.1846153846153846
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.2802064293355139
	Recall	0.7641435485352097
	F1 Score	0.24779643066933066
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
Weighted Avg	Precision	0.9917542211368887
	Recall	0.6221151560274766

	F1 Score	0.7610298468001963
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
	Accuracy	0.6221151560274766
	True Positives	16327
	True Negatives	2756427
	False Positives	942748
	False Negatives	4299
	AUC	0.66
	Duration (seconds)	26.606997

Table C.99: Word2vec 9 Classification Results on 25_filtered_chunk_extraction_-e_none_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.999407440557586
	Recall	0.7609417881769869
	F1 Score	0.8640227433529812
	Support	4437346.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	24506437
1.0	Precision	0.07188114485470833
	Recall	0.9248995983935743
	F1 Score	0.13339511714790467
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.003923503160599768
	Recall	0.8240963855421687
	F1 Score	0.007809823936425293
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.23386892712550608
	Recall	0.9898259705488621
	F1 Score	0.3783451875351789
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.3272702539246001
	Recall	0.8749409356653981
	F1 Score	0.3458932179931225

	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
Weighted Avg	Precision	0.9932820994307777
	Recall	0.7620084589407387
	F1 Score	0.8592355832611801
	Support	4467226.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	24674149
	Accuracy	0.7620084589407387
	True Positives	20727
	True Negatives	3376562
	False Positives	267572
	False Negatives	1436
	AUC	0.76
	Duration (seconds)	20.313128

C.5.2 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Table C.100: Transformers 0 Classification Results on 26_-filtered_chunk_extraction_-e_only-max-entropy_-s_none

Class	Metric Name	Metric Value
0.0	Precision	0.9991307725028203
	Recall	0.9804896640592389
	F1 Score	0.9897224512228635
	Support	110198.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	620669
1.0	Precision	0.9121229461293223
	Recall	0.9958054439982151
	F1 Score	0.9521290212475467
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	1.0
	Recall	1.0
	F1 Score	1.0
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	1.0
	Recall	1.0

	F1 Score	1.0
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9778134296580356
	Recall	0.9940737770143635
	F1 Score	0.9854628681176025
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
Weighted Avg	Precision	0.9852574143764466
	Recall	0.9839803538028813
	F1 Score	0.9842562432788491
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
	Accuracy	0.9839803538028813
	True Positives	22316
	True Negatives	108048
	False Positives	2150
	False Negatives	94
	AUC	0.93
	Duration (seconds)	1.381033

Table C.101: Transformers 1 Classification Results on 26_-filtered_chunk_extraction_-e_only-max-entropy_-s_none

Class	Metric Name	Metric Value
0.0	Precision	0.9989467459994826
	Recall	0.9811611825985953
	F1 Score	0.9899740882829596
	Support	110198.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	620669
1.0	Precision	0.9148202855736091
	Recall	0.9949129852744311
	F1 Score	0.953187123252533
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	1.0
	Recall	1.0
	F1 Score	1.0

	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	1.0
	Recall	1.0
	F1 Score	1.0
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9784417578932729
	Recall	0.9940185419682566
	F1 Score	0.9857903028838731
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
Weighted Avg	Precision	0.985544169072628
	Recall	0.9843658533102986
	F1 Score	0.9846234812939566
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
	Accuracy	0.9843658533102986
	True Positives	22296
	True Negatives	108122
	False Positives	2076
	False Negatives	114
	AUC	0.93
	Duration (seconds)	1.309276

Table C.102: Word2vec 0 Classification Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Class	Metric Name	Metric Value
0.0	Precision	0.9987721021611002
	Recall	0.9595636944409155
	F1 Score	0.978775396862128
	Support	110198.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	620669
1.0	Precision	0.8391959798994975
	Recall	0.991120035698349
	F1 Score	0.908852834666612
	Support	22410.0

	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.9744272775705913
	Recall	0.9793842034805891
	F1 Score	0.9768994525303779
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9307402760351318
	Recall	0.9930388219544846
	F1 Score	0.960880829015544
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9357839089165803
	Recall	0.9807766888935845
	F1 Score	0.9563521282686656
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
Weighted Avg	Precision	0.9707796430289842
	Recall	0.966033210068676
	F1 Score	0.9670618695288739
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
	Accuracy	0.966033210068676
	True Positives	22211
	True Negatives	105742
	False Positives	4245
	False Negatives	129
	AUC	0.91
	Duration (seconds)	1.785186

Table C.103: Word2vec 1 Classification Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Class	Metric Name	Metric Value
0.0	Precision	0.9991360772271836
	Recall	0.9655256901214178
	F1 Score	0.9820433893737107
	Support	110198.0
	Final Samples (after rebalancing)	20964

	Initial Samples (before rebalancing)	620669
1.0	Precision	0.8559530206494205
	Recall	0.9951360999553771
	F1 Score	0.9203119841531859
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.9854766305782942
	Recall	0.9991967871485944
	F1 Score	0.9922892847646901
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9970635344367326
	Recall	1.0
	F1 Score	0.9985296083411309
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9594073157229077
	Recall	0.9899646443063473
	F1 Score	0.9732935666581793
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
Weighted Avg	Precision	0.9758098498505534
	Recall	0.972079841231314
	F1 Score	0.972880234960717
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
	Accuracy	0.972079841231314
	True Positives	22301
	True Negatives	106399
	False Positives	3750
	False Negatives	92
	AUC	0.92
	Duration (seconds)	1.557051

Table C.104: Word2vec 2 Classification Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Class	Metric Name	Metric Value

0.0	Precision	0.9988499462679814
	Recall	0.96154195175956
	F1 Score	0.9798409469206584
	Support	110198.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	620669
1.0	Precision	0.8443304843304843
	Recall	0.9918340026773762
	F1 Score	0.9121575869498308
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.9769169540992305
	Recall	0.985809906291834
	F1 Score	0.9813432835820894
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9505381855458739
	Recall	0.9930388219544846
	F1 Score	0.9713238182532408
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9426588925608925
	Recall	0.9830561706708136
	F1 Score	0.9611664089264548
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
Weighted Avg	Precision	0.9722565818990822
	Recall	0.9678750410485587
	F1 Score	0.9688257671987277
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
	Accuracy	0.9678750410485587
	True Positives	22227
	True Negatives	105960
	False Positives	4096
	False Negatives	122
	AUC	0.92
	Duration (seconds)	9.970666

Table C.105: Word2vec 3 Classification Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Class	Metric Name	Metric Value
0.0	Precision	0.9992522692209402
	Recall	0.9580391658650792
	F1 Score	0.9782118220439099
	Support	110198.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	620669
1.0	Precision	0.8289766820139611
	Recall	0.9962516733601071
	F1 Score	0.9049491305581452
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.9936136242682277
	Recall	0.9997322623828648
	F1 Score	0.9966635526491391
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	1.0
	Recall	1.0
	F1 Score	1.0
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9554606438757822
	Recall	0.9885057754020128
	F1 Score	0.9699561263127986
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
Weighted Avg	Precision	0.9718807799524827
	Recall	0.9663830151772583
	F1 Score	0.9675640339706976
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
	Accuracy	0.9663830151772583
	True Positives	22326
	True Negatives	105574
	False Positives	4605

	False Negatives	79
	AUC	0.92
	Duration (seconds)	3.241017

Table C.106: Word2vec 4 Classification Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Class	Metric Name	Metric Value
0.0	Precision	0.9977117150452302
	Recall	0.9297990889126845
	F1 Score	0.9625590079616712
	Support	110198.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	620669
1.0	Precision	0.7428845758178452
	Recall	0.9667112896028559
	F1 Score	0.8401458155588305
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.8468968848104322
	Recall	0.9389558232931727
	F1 Score	0.8905535804977146
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9060814124570868
	Recall	0.9892904953145917
	F1 Score	0.9458594649942403
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.8733936470326487
	Recall	0.9561891742808262
	F1 Score	0.9097794672531142
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
Weighted Avg	Precision	0.9504793961858847
	Recall	0.9375348020388641
	F1 Score	0.9406098602988769
	Support	140078.0
	Final Samples (after rebalancing)	83856

	Initial Samples (before rebalancing)	788381
	Accuracy	0.9375348020388641
	True Positives	21664
	True Negatives	102462
	False Positives	7375
	False Negatives	234
	AUC	0.9
	Duration (seconds)	2.122273

Table C.107: Word2vec 5 Classification Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Class	Metric Name	Metric Value
0.0	Precision	0.9983198956270356
	Recall	0.9651899308517395
	F1 Score	0.981475415130641
	Support	110198.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	620669
1.0	Precision	0.8553068313392512
	Recall	0.9888888888888889
	F1 Score	0.9172599337748344
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.9586437194965323
	Recall	0.9991967871485944
	F1 Score	0.9785002621919245
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9994643813604713
	Recall	0.9991967871485944
	F1 Score	0.9993305663408756
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9529337069558227
	Recall	0.9881180985094543
	F1 Score	0.9691415443595689
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381

Weighted Avg	Precision	0.974412939257568
	Recall	0.9707948428732563
	F1 Score	0.9715988310586275
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
	Accuracy	0.9707948428732563
	True Positives	22161
	True Negatives	106362
	False Positives	3747
	False Negatives	178
	AUC	0.92
	Duration (seconds)	1.741212

Table C.108: Word2vec 6 Classification Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Class	Metric Name	Metric Value
0.0	Precision	0.9978788751319705
	Recall	0.9434744732209296
	F1 Score	0.9699143608784073
	Support	110198.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	620669
1.0	Precision	0.7849791276810134
	Recall	0.9733601070950468
	F1 Score	0.8690784493406113
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.8817042606516291
	Recall	0.94190093708166
	F1 Score	0.9108090614886732
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9
	Recall	0.9903614457831326
	F1 Score	0.9430210325047802
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.8911405658661532

	Recall	0.9622742407951922
	F1 Score	0.923205726053118
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
Weighted Avg	Precision	0.9581112233659691
	Recall	0.9494638701294993
	F1 Score	0.9514893572928468
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
	Accuracy	0.9494638701294993
	True Positives	21813
	True Negatives	103969
	False Positives	5891
	False Negatives	221
	AUC	0.91
	Duration (seconds)	1.910313

Table C.109: Word2vec 7 Classification Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Class	Metric Name	Metric Value
0.0	Precision	0.9991264965866905
	Recall	0.9549265866894135
	F1 Score	0.9765266493752348
	Support	110198.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	620669
1.0	Precision	0.819746276889134
	Recall	0.9947791164658635
	F1 Score	0.8988206833988509
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.9807945277558537
	Recall	0.9981258366800535
	F1 Score	0.9893842887473461
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9909550412343708
	Recall	0.9973226238286479

	F1 Score	0.9941286362423272
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9476555856165123
	Recall	0.9862885409159946
	F1 Score	0.9647150644409397
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
Weighted Avg	Precision	0.9697221574845738
	Recall	0.9635845743085995
	F1 Score	0.96490723737958
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
	Accuracy	0.9635845743085995
	True Positives	22293
	True Negatives	105231
	False Positives	4900
	False Negatives	92
	AUC	0.91
	Duration (seconds)	1.76205

Table C.110: Word2vec 8 Classification Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Class	Metric Name	Metric Value
0.0	Precision	0.9971711734718934
	Recall	0.9212599139730303
	F1 Score	0.9577136603980057
	Support	110198.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	620669
1.0	Precision	0.7212570693216838
	Recall	0.9503792949576082
	F1 Score	0.8201159051964804
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.7426787252368647
	Recall	0.9234270414993306
	F1 Score	0.8232485976846879

	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.904541015625
	Recall	0.9919678714859438
	F1 Score	0.9462393053249905
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.8414119959138604
	Recall	0.9467585304789783
	F1 Score	0.8868293671510411
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
Weighted Avg	Precision	0.9437742231462023
	Recall	0.9278616199545967
	F1 Score	0.9318091684756615
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
	Accuracy	0.9278616199545967
	True Positives	21298
	True Negatives	101521
	False Positives	7995
	False Negatives	288
	AUC	0.89
	Duration (seconds)	1.797068

Table C.111: Word2vec 9 Classification Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Class	Metric Name	Metric Value
0.0	Precision	0.9976229500340263
	Recall	0.9445089747545328
	F1 Score	0.9703396743563808
	Support	110198.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	620669
1.0	Precision	0.7953229076165653
	Recall	0.9803659080767515
	F1 Score	0.8782028220809849
	Support	22410.0

	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.8817815683487326
	Recall	0.9965194109772423
	F1 Score	0.9356460532931121
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9564325986673501
	Recall	0.9991967871485944
	F1 Score	0.9773471258347518
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9077900061666686
	Recall	0.9801477702392802
	F1 Score	0.9403839188913073
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
Weighted Avg	Precision	0.9610714753304725
	Recall	0.9530904210511286
	F1 Score	0.9548611930610224
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
	Accuracy	0.9530904210511286
	True Positives	21970
	True Negatives	104083
	False Positives	5649
	False Negatives	245
	AUC	0.9
	Duration (seconds)	1.753935

Table C.112: Word2vec 10 Classification Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Class	Metric Name	Metric Value
0.0	Precision	0.9968879153988721
	Recall	0.9127479627579448
	F1 Score	0.9529643051706579
	Support	110198.0
	Final Samples (after rebalancing)	20964

	Initial Samples (before rebalancing)	620669
1.0	Precision	0.6930385946581337
	Recall	0.9471218206157965
	F1 Score	0.8003997284863112
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.7735207435955566
	Recall	0.9135207496653279
	F1 Score	0.8377117603731893
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.893581081081081
	Recall	0.9914323962516733
	F1 Score	0.9399670008884375
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.8392570836834109
	Recall	0.9412057323226856
	F1 Score	0.882760698729649
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
Weighted Avg	Precision	0.9395670606560695
	Recall	0.9203657961992604
	F1 Score	0.925137056424896
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
	Accuracy	0.9203657961992604
	True Positives	21225
	True Negatives	100583
	False Positives	9225
	False Negatives	313
	AUC	0.89
	Duration (seconds)	1.967521

Table C.113: Word2vec 11 Classification Results on 26_filtered_chunk_extraction_-e_only-max-entropy_-s_none

Class	Metric Name	Metric Value

0.0	Precision	0.9982958061598515
	Recall	0.9515236211183506
	F1 Score	0.9743487290517718
	Support	110198.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	620669
1.0	Precision	0.8143036386449184
	Recall	0.9846497099509147
	F1 Score	0.8914114890522743
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.8951184146930884
	Recall	0.9917001338688086
	F1 Score	0.9409373809221389
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9797741003414763
	Recall	0.998661311914324
	F1 Score	0.9891275523733757
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9218729899598337
	Recall	0.9816336942130995
	F1 Score	0.9489562878498902
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
Weighted Avg	Precision	0.9656153666734965
	Recall	0.9591513299733005
	F1 Score	0.9605834266591995
	Support	140078.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	788381
	Accuracy	0.9591513299733005
	True Positives	22066
	True Negatives	104856
	False Positives	5021
	False Negatives	178
	AUC	0.91
	Duration (seconds)	2.024159

C.5.3 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Table C.114: Transformers 0 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9979907885382214
	Recall	0.9637755787399812
	F1 Score	0.9805848095306794
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.9065070445476016
	Recall	0.9933958054439982
	F1 Score	0.9479645716232329
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.9878177966101694
	Recall	0.998661311914324
	F1 Score	0.9932099587272
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9718969555035128
	Recall	1.0
	F1 Score	0.9857482185273159
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9660531462998764
	Recall	0.9889581740245759
	F1 Score	0.9768768896021071
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.9754306125035215
	Recall	0.9733688415446072
	F1 Score	0.9737249197026006
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.9733688415446072
	True Positives	22262

	True Negatives	64572
	False Positives	2294
	False Negatives	127
	AUC	0.88
	Duration (seconds)	1.353184

Table C.115: Transformers 1 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9981469570277803
	Recall	0.9728055642621531
	F1 Score	0.9853133479973091
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.9244328314877027
	Recall	0.9946006247211067
	F1 Score	0.958233915865953
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.9997323340471093
	Recall	1.0
	F1 Score	0.9998661491098916
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	1.0
	Recall	0.9997322623828648
	F1 Score	0.9998661132681751
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9805780306406481
	Recall	0.9917846128415311
	F1 Score	0.9858198815603322
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.9812280060199797
	Recall	0.9799337317684947
	F1 Score	0.9801714618958681

	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.9799337317684947
	True Positives	22289
	True Negatives	65177
	False Positives	1822
	False Negatives	121
	AUC	0.89
	Duration (seconds)	1.23036

Table C.116: Transformers 2 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9724309160674556
	Recall	0.9002522425707846
	F1 Score	0.9349505909707421
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.7653359683794466
	Recall	0.8640339134315038
	F1 Score	0.8116956612869419
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.9983952928590533
	Recall	0.9994645247657296
	F1 Score	0.9989296226919989
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.5810113519091847
	Recall	0.9044176706827309
	F1 Score	0.7075086396481306
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.829293382303785
	Recall	0.9170420878626873
	F1 Score	0.8632711286494534
	Support	96879.0

	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.9104363362049178
	Recall	0.895859783854086
	F1 Score	0.9001372983177933
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.895859783854086
	True Positives	19363
	True Negatives	60316
	False Positives	5633
	False Negatives	1657
	AUC	0.83
	Duration (seconds)	1.075649

Table C.117: Transformers 3 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9589999797730536
	Recall	0.7076523530201944
	F1 Score	0.814373314553668
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.5099574681010758
	Recall	0.9095493083444891
	F1 Score	0.6535107406219942
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	1.0
	Recall	1.0
	F1 Score	1.0
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	1.0
	Recall	1.0
	F1 Score	1.0
	Support	3735.0
	Final Samples (after rebalancing)	20964

	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.8672393619685324
	Recall	0.9043004153411709
	F1 Score	0.8669710137939155
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.8582890668252142
	Recall	0.7768969539322247
	F1 Score	0.7914756902849955
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.7768969539322247
	True Positives	20383
	True Negatives	47412
	False Positives	19587
	False Negatives	2027
	AUC	0.73
	Duration (seconds)	1.191172

Table C.118: Transformers 4 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.998817222469701
	Recall	0.9705219480887775
	F1 Score	0.9844663133989403
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.9187510284679941
	Recall	0.9965640339134315
	F1 Score	0.9560768868530332
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	1.0
	Recall	1.0
	F1 Score	1.0
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964

4.0	Precision	1.0
	Recall	1.0
	F1 Score	1.0
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9793920627344237
	Recall	0.9917714955005522
	F1 Score	0.9851358000629934
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.9803875518555645
	Recall	0.9788189390889667
	F1 Score	0.9790970340919299
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.9788189390889667
	True Positives	22333
	True Negatives	65024
	False Positives	1975
	False Negatives	77
	AUC	0.88
	Duration (seconds)	1.394558

Table C.119: Transformers 5 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9272212406515478
	Recall	0.5717995790981955
	F1 Score	0.707374718416485
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.403455876237212
	Recall	0.8658188308790719
	F1 Score	0.5504241014439306
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.9547546012269938

	Recall	1.0
	F1 Score	0.9768536681051393
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	1.0
	Recall	0.9526104417670683
	F1 Score	0.9757301522007406
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.8213579295289384
	Recall	0.8475572129360839
	F1 Score	0.8025956600415739
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.8099314663081967
	Recall	0.6710019715314981
	F1 Score	0.6918042448971091
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.6710019715314981
	True Positives	19403
	True Negatives	38310
	False Positives	28689
	False Negatives	3007
	AUC	0.65
	Duration (seconds)	1.382089

Table C.120: Transformers 6 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9376038584687715
	Recall	0.6905625457096375
	F1 Score	0.7953414414027247
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.4825279552715655
	Recall	0.8626506024096385

	F1 Score	0.618881454685149
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	1.0
	Recall	0.9997322623828648
	F1 Score	0.9998661132681751
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	1.0
	Recall	1.0
	F1 Score	1.0
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.8550329534350842
	Recall	0.8882363526255352
	F1 Score	0.8535222523390122
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.8371470844164887
	Recall	0.7542191806273806
	F1 Score	0.7702981509418139
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.7542191806273806
	True Positives	19332
	True Negatives	46267
	False Positives	20732
	False Negatives	3078
	AUC	0.7
	Duration (seconds)	0.944381

Table C.121: Transformers 7 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.981802328397733
	Recall	0.9075359333721399
	F1 Score	0.9432094935236175

	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.777663444930359
	Recall	0.9442659526996876
	F1 Score	0.8529050200519941
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.8948166877370417
	Recall	0.9475234270414993
	F1 Score	0.9204161248374513
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9632469592808038
	Recall	0.9753681392235609
	F1 Score	0.9692696554476521
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9043823550864843
	Recall	0.943673363084222
	F1 Score	0.9214500734651787
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.9305120792206845
	Recall	0.9201891018693422
	F1 Score	0.9224462550740501
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.9201891018693422
	True Positives	21161
	True Negatives	60804
	False Positives	6007
	False Negatives	1113
	AUC	0.84
	Duration (seconds)	0.980153

Table C.122: Word2vec 0 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9997161689110348
	Recall	0.9462827803400051
	F1 Score	0.9722658855824011
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.8548779530649085
	Recall	0.97206604194556
	F1 Score	0.9097135220913722
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.8395209580838323
	Recall	0.9384203480589023
	F1 Score	0.8862199747155499
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.972397476340694
	Recall	0.9903614457831326
	F1 Score	0.9812972542777556
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9166281391001174
	Recall	0.9617826540319
	F1 Score	0.9373741591667697
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.9589829981898101
	Recall	0.9536432044096244
	F1 Score	0.9548271446700827
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.9536432044096244
	True Positives	21784
	True Negatives	63400
	False Positives	3572

	False Negatives	18
	AUC	0.88
	Duration (seconds)	1.90141

Table C.123: Word2vec 1 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.999862999101883
	Recall	0.9803728413856924
	F1 Score	0.9900220058481295
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.9444725524917784
	Recall	0.9995983935742971
	F1 Score	0.9712539021852237
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.999732118939191
	Recall	0.9991967871485944
	F1 Score	0.9994643813604713
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	1.0
	Recall	1.0
	F1 Score	1.0
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9860169176332132
	Recall	0.994792005527146
	F1 Score	0.9901850723484561
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.9870503457137841
	Recall	0.9863025010580209
	F1 Score	0.9864292961546988
	Support	96879.0
	Final Samples (after rebalancing)	83856

	Initial Samples (before rebalancing)	543872
	Accuracy	0.9863025010580209
	True Positives	22401
	True Negatives	65684
	False Positives	1315
	False Negatives	8
	AUC	0.89
	Duration (seconds)	1.210697

Table C.124: Word2vec 2 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9995303326810177
	Recall	0.9529246705174704
	F1 Score	0.9756712563228754
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.8668831168831169
	Recall	0.941231593038822
	F1 Score	0.9025287749775364
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.6843073150796894
	Recall	0.8966532797858099
	F1 Score	0.7762197241858849
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9812069878242456
	Recall	0.9925033467202142
	F1 Score	0.9868228404099562
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.8829819381170174
	Recall	0.9458282225155792
	F1 Score	0.9103106489740631
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872

Weighted Avg	Precision	0.9559871523239541
	Recall	0.9495762755602349
	F1 Score	0.9514924011229307
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.9495762755602349
	True Positives	21093
	True Negatives	63845
	False Positives	2929
	False Negatives	25
	AUC	0.88
	Duration (seconds)	1.46971

Table C.125: Word2vec 3 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9997874343323919
	Recall	0.9828206391140166
	F1 Score	0.9912314373668721
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.9511148863877681
	Recall	0.9992860330209727
	F1 Score	0.9746055924273745
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.9991972170189992
	Recall	0.9997322623828648
	F1 Score	0.9994646680942185
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	1.0
	Recall	1.0
	F1 Score	1.0
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9875248844347898

	Recall	0.9954597336294635
	F1 Score	0.9913254244721164
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.9885139661056759
	Recall	0.9879437236139927
	F1 Score	0.9880410298802881
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.9879437236139927
	True Positives	22394
	True Negatives	65848
	False Positives	1150
	False Negatives	14
	AUC	0.89
	Duration (seconds)	1.442202

Table C.126: Word2vec 4 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9997476858057496
	Recall	0.9462380035522918
	F1 Score	0.9722571542496089
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.8488494999593463
	Recall	0.9317269076305221
	F1 Score	0.8883594281824372
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.6574074074074074
	Recall	0.8934404283801874
	F1 Score	0.7574622630802408
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9781118143459916
	Recall	0.9930388219544846

	F1 Score	0.9855187989903017
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.8710291018796238
	Recall	0.9411110403793714
	F1 Score	0.9008994111256471
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.9508095334245533
	Recall	0.9426501099309448
	F1 Score	0.9450802148037392
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.9426501099309448
	True Positives	20880
	True Negatives	63397
	False Positives	3401
	False Negatives	16
	AUC	0.88
	Duration (seconds)	1.80203

Table C.127: Word2vec 5 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.999649854613546
	Recall	0.980074329467604
	F1 Score	0.9897653105828799
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.9435497470489039
	Recall	0.9987059348505132
	F1 Score	0.9703446780836765
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.9981268397109981
	Recall	0.998661311914324
	F1 Score	0.9983940042826552

	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	1.0
	Recall	1.0
	F1 Score	1.0
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.985331610343362
	Recall	0.9943603940581103
	F1 Score	0.9896259982373029
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.9866275889195749
	Recall	0.9858689705715377
	F1 Score	0.9860001846178559
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.9858689705715377
	True Positives	22381
	True Negatives	65664
	False Positives	1335
	False Negatives	22
	AUC	0.89
	Duration (seconds)	1.441652

Table C.128: Word2vec 6 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.999731297516873
	Recall	0.9440439409543426
	F1 Score	0.9710899236945941
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.8525708289611752
	Recall	0.9426595269968764
	F1 Score	0.8953547512079343
	Support	22410.0

	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.6763477222995823
	Recall	0.9103078982597055
	F1 Score	0.7760785208856426
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.972681901759916
	Recall	0.9914323962516733
	F1 Score	0.9819676478387696
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.8753329376343866
	Recall	0.9471109406156494
	F1 Score	0.9061227109067351
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.9521798854779173
	Recall	0.9442500438691563
	F1 Score	0.9464719517374208
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.9442500438691563
	True Positives	21125
	True Negatives	63250
	False Positives	3415
	False Negatives	17
	AUC	0.88
	Duration (seconds)	1.769334

Table C.129: Word2vec 7 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9997866536626995
	Recall	0.9792235705010522
	F1 Score	0.9893982808022923
	Support	66999.0
	Final Samples (after rebalancing)	20964

	Initial Samples (before rebalancing)	376160
1.0	Precision	0.9464058928117857
	Recall	0.9975903614457832
	F1 Score	0.9713242961418145
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.9566888774987186
	Recall	0.9994645247657296
	F1 Score	0.9776090087730784
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	1.0
	Recall	0.9997322623828648
	F1 Score	0.9998661132681751
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.975720355993301
	Recall	0.9940026797738575
	F1 Score	0.9845494247463401
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.9857853097587407
	Recall	0.9850431982163317
	F1 Score	0.9851664702653303
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.9850431982163317
	True Positives	22356
	True Negatives	65607
	False Positives	1265
	False Negatives	13
	AUC	0.89
	Duration (seconds)	1.426085

Table C.130: Word2vec 8 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value

0.0	Precision	0.9992136219929065
	Recall	0.9292974522007791
	F1 Score	0.9629881679684479
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.8176832034310694
	Recall	0.8847835787594823
	F1 Score	0.849911056816477
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.48869752421959095
	Recall	0.8508701472556894
	F1 Score	0.6208243797616723
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9664570230607966
	Recall	0.9874163319946453
	F1 Score	0.9768242616871937
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.8180128431760908
	Recall	0.913091877552649
	F1 Score	0.8526369665584478
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.9362771734110101
	Recall	0.9182175703712879
	F1 Score	0.9241731306556302
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.9182175703712879
	True Positives	19828
	True Negatives	62262
	False Positives	3996
	False Negatives	44
	AUC	0.87
	Duration (seconds)	1.689776

Table C.131: Word2vec 9 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9998785412807822
	Recall	0.9829698950730608
	F1 Score	0.9913521243367328
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.9519386450788241
	Recall	0.9969656403391343
	F1 Score	0.9739319965126417
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.9749604638903532
	Recall	0.9903614457831326
	F1 Score	0.9826006109709126
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9930648172846093
	Recall	0.9967871485943776
	F1 Score	0.9949225013361839
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9799606168836422
	Recall	0.9917710324474263
	F1 Score	0.9857018082891178
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.9875657454007171
	Recall	0.987025051868826
	F1 Score	0.9871227597802626
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.987025051868826
	True Positives	22342
	True Negatives	65858
	False Positives	1119

	False Negatives	7
	AUC	0.89
	Duration (seconds)	1.566643

Table C.132: Word2vec 10 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9989445827163606
	Recall	0.9465066642785713
	F1 Score	0.972018914631249
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.8467150293044797
	Recall	0.9218652387327086
	F1 Score	0.8826934991134183
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.6321662177760677
	Recall	0.8797858099062918
	F1 Score	0.7356990932497482
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9718421052631578
	Recall	0.9887550200803212
	F1 Score	0.980225613802256
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.8624169837650164
	Recall	0.9342281832494733
	F1 Score	0.892659280199168
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
Weighted Avg	Precision	0.9485455360880928
	Recall	0.9398631282321246
	F1 Score	0.9425617043667872
	Support	96879.0
	Final Samples (after rebalancing)	83856

	Initial Samples (before rebalancing)	543872
	Accuracy	0.9398631282321246
	True Positives	20659
	True Negatives	63415
	False Positives	3406
	False Negatives	59
	AUC	0.88
	Duration (seconds)	1.67752

Table C.133: Word2vec 11 Classification Results on 27_filtered_chunk_extraction_-e_only-max-entropy_-s_activate

Class	Metric Name	Metric Value
0.0	Precision	0.9997263772345859
	Recall	0.9815967402498544
	F1 Score	0.9905786132260906
	Support	66999.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	376160
1.0	Precision	0.9518972213922916
	Recall	0.9951807228915662
	F1 Score	0.973057876480726
	Support	22410.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	125784
2.0	Precision	0.9452582883577486
	Recall	0.9847389558232932
	F1 Score	0.964594807238395
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
4.0	Precision	0.9872847682119206
	Recall	0.9978580990629183
	F1 Score	0.99254327563249
	Support	3735.0
	Final Samples (after rebalancing)	20964
	Initial Samples (before rebalancing)	20964
Macro Avg	Precision	0.9710416637991367
	Recall	0.989843629506908
	F1 Score	0.9801936431444255
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872

Weighted Avg	Precision	0.9860829756296925
	Recall	0.985487050857255
	F1 Score	0.9855997095241555
	Support	96879.0
	Final Samples (after rebalancing)	83856
	Initial Samples (before rebalancing)	543872
	Accuracy	0.985487050857255
	True Positives	22302
	True Negatives	65766
	False Positives	1118
	False Negatives	18
	AUC	0.89
	Duration (seconds)	1.698716

Acronyms

BFD Byte Frequency Distribution. 5–7

CNN Convolutional Neural Networks. 5, 8, 10, 11

GRU Gated Recurrent Units. 5, 9

KNN K-Nearest Neighbors. 17

LSB Least Significant Bit. 25

LSTM Long Short-Term Memory. 5, 9, 10

OPTICS Ordering Points To Identify the Clustering Structure. 19–21

PCA Principal Component Analysis. 15

RCNN Recurrent Convolutional Neural Network. 8

REGEX regular expressions. 35

RNN Recurrent Neural Networks. 5, 8–10

Seq2Seq Sequence-to-Sequence. 10, 11

SMOTE Synthetic Minority Over-sampling Technique. 16

SSH Secure Shell. 1

SVM Support Vector Machines. 17

t-SNE t-Distributed Stochastic Neighbor Embedding. 15

VMI Virtual Machine Introspection. 1, 2

Glossary

chunk In our study, chunks are defined as a series of bytes that are allocated in the heap. These structures are allocated using the `malloc` function and begin everytime by a *malloc header*. v, 24–27

pointer In our study, pointers are characterized as sequences of hexadecimal numbers that reference distinct memory addresses. These sequences can be recognized using the following regular expression: `"[0-9a-f]{12}0{4}"`. 31, 34–37

. 31, 34–37, 39

value node In our study, value nodes represent 8-byte blocks of data that are contained within a structure.. 35, 36

References

[1] Mihael Ankerst et al. „OPTICS: Ordering Points To Identify the Clustering Structure“. In: *ACM SIGMOD Record* 28 (June 1999), pp. 49–60. DOI: 10.1145/304181.304187.

[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. Apr. 19, 2018. arXiv: 1803.01271[cs]. URL: http://arxiv.org/abs/1803.01271 (visited on 08/23/2023).

[3] Richard Boddy and Gordon Smith. *Statistical methods in practice: for scientists and technologists*. John Wiley & Sons, 2009.

[4] Meghan K. Cain, Zhiyong Zhang, and Ke-Hai Yuan. „Univariate and multivariate skewness and kurtosis for measuring nonnormality: Prevalence, influence and estimation“. In: *Behavior Research Methods* 49.5 (Oct. 2017), pp. 1716–1735. ISSN: 1554-3528. DOI: 10.3758/s13428-016-0814-1. URL: http://link.springer.com/10.3758/s13428-016-0814-1 (visited on 08/30/2023).

[5] Junyoung Chung et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. Dec. 11, 2014. arXiv: 1412.3555[cs]. URL: http://arxiv.org/abs/1412.3555 (visited on 08/23/2023).

[6] D. J. Delorie et al. *Malloc Internals*. Publisher: Sourceware. 2023. URL: https://sourceware.org/glibc/wiki/MallocInternals (visited on 09/25/2023).

[7] John Ellson et al. „Graphviz and Dynagraph — Static and Dynamic Graph Drawing Tools“. In: *Graph Drawing Software*. Ed. by Michael Jünger and Petra Mutzel. Red. by Gerald Farin et al. Series Title: Mathematics and Visualization. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 127–148. ISBN: 978-3-642-62214-4 978-3-642-18638-7. DOI: 10.1007/978-3-642-18638-7_6. URL: http://link.springer.com/10.1007/978-3-642-18638-7_6 (visited on 09/11/2023).

[8] Martin Ester et al. „A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise“. In: *KDD-96* (Aug. 2, 1996).

[9] Christofer Fellicious et al. *SmartKex: Machine Learning Assisted SSH Keys Extraction From The Heap Dump*. Sept. 13, 2022. arXiv: 2209.05243[cs]. URL: http://arxiv.org/abs/2209.05243 (visited on 08/17/2023).

[10] Jonas Gehring et al. „Convolutional Sequence to Sequence Learning“. In: *Facebook AI Research* (July 25, 2017). URL: https://arxiv.org/pdf/1705.03122.pdf.

[11] Wolfram Gloger and Doug Lea. *Malloc implementation for multiple threads without lock contention*. Publisher: Free Software Foundation, Inc. 2001. URL: https://elixir.bootlin.com/glibc/glibc-2.28/source/malloc/malloc.c (visited on 09/22/2023).

[12] Luke Hiester. „File Fragment Classification Using Neural Networks with Lossless Representations“. In: *East Tennessee State University* (May 2018). (Visited on 08/21/2023).

[13] G. E. Hinton and R. R. Salakhutdinov. „Reducing the Dimensionality of Data with Neural Networks“. In: *Science* 313.5786 (July 28, 2006), pp. 504–507. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1127647. URL: https://www.science.org/doi/10.1126/science.1127647 (visited on 08/30/2023).

[14] Sepp Hochreiter and Jürgen Schmidhuber. „Long short-term memory“. In: *Neural computation* 9.8 (1997). Publisher: MIT Press, pp. 1735–1780. (Visited on 08/23/2023).

[15] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. „A survey of feature selection and feature extraction techniques in machine learning“. In: *2014 Science and Information Conference*. 2014 Science and Information Conference. Aug. 2014, pp. 372–378. DOI: 10.1109/SAI.2014.6918213.

[16] Udayan Khurana, Horst Samulowitz, and Deepak Turaga. *Feature Engineering for Predictive Modeling using Reinforcement Learning*. Sept. 21, 2017. arXiv: 1709.07150[cs,stat]. URL: http://arxiv.org/abs/1709.07150 (visited on 08/30/2023).

[17] Mario Koppen. „The curse of dimensionality“. In: 1 (2000), pp. 4–8.

[18] S. B. Kotsiantis. „Decision trees: a recent overview“. In: *Artificial Intelligence Review* 39.4 (Apr. 2013), pp. 261–283. ISSN: 0269-2821, 1573-7462. DOI: 10.1007/s10462-011-9272-4. URL: http://link.springer.com/10.1007/s10462-011-9272-4 (visited on 08/30/2023).

[19] J. Laaksonen and E. Oja. „Classification with learning k-nearest neighbors“. In: *Proceedings of International Conference on Neural Networks (ICNN'96)*. International Conference on Neural Networks (ICNN'96). Vol. 3. Washington, DC, USA: IEEE, 1996, pp. 1480–1483. ISBN: 978-0-7803-3210-2. DOI: 10.1109/ICNN.1996.549118. URL: http://ieeexplore.ieee.org/document/549118/ (visited on 08/30/2023).

[20] Siwei Lai et al. „Recurrent Convolutional Neural Networks for Text Classification“. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 29.1 (Feb. 19, 2015). ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v29i1.9513. URL: https://ojs.aaai.org/index.php/AAAI/article/view/9513 (visited on 08/23/2023).

[21] Yann LeCun et al. „Gradient-Based Learning Applied to Document Recognition“. In: *proc of the IEEE* (1998).

[22] Ulrike von Luxburg. *A Tutorial on Spectral Clustering*. Nov. 1, 2007. arXiv: 0711.0189[cs]. URL: http://arxiv.org/abs/0711.0189 (visited on 09/05/2023).

[23] J Macqueen. „SOME METHODS FOR CLASSIFICATION AND ANALYSIS OF MULTIVARIATE OBSERVATIONS“. In: *MULTIVARIATE OBSERVATIONS* VOL. 5.1 (1967).

[24] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. Sept. 6, 2013. arXiv: 1301.3781[cs]. URL: http://arxiv.org/abs/1301.3781 (visited on 10/18/2023).

[25] Todd G Nick and Kathleen M Campbell. „Logistic regression“. In: *Topics in biostatistics* (2007). Publisher: Springer, pp. 273–301.

[26] F. Pedregosa et al. „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[27] Philipp Probst, Marvin Wright, and Anne-Laure Boulesteix. „Hyperparameters and Tuning Strategies for Random Forest“. In: *WIREs Data Mining and Knowledge Discovery* 9.3 (May 2019), e1301. ISSN: 1942-4787, 1942-4795. DOI: 10.1002/widm.1301. arXiv: 1804.03515[cs,stat]. URL: http://arxiv.org/abs/1804.03515 (visited on 08/30/2023).

[28] Dr D Ramyachitra and P Manikandan. „IMBALANCED DATASET CLASSIFICATION AND SOLUTIONS: A REVIEW“. In: *International Journal of Computing and Business Research* 5.4 (2014).

[29] Radim Řehůřek and Petr Sojka. „Software Framework for Topic Modelling with Large Corpora“. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 22, 2010, pp. 45–50.

[30] Stewart Sentanoe and Hans P. Reiser. „SSHkex: Leveraging virtual machine introspection for extracting SSH keys and decrypting SSH network traffic“. In: *Forensic Science International: Digital Investigation* 40 (Apr. 2022), p. 301337. ISSN: 26662817. DOI: 10.1016/j.fsidi.2022.301337. URL: https://linkinghub.elsevier.com/retrieve/pii/S2666281722000063 (visited on 08/17/2023).

[31] C E Shannon. „A Mathematical Theory of Communication“. In: *The Bell System Technical Journa* 27 (Oct. 1948), pp. 379–423.

[32] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. *Sequence to Sequence Learning with Neural Networks*. Dec. 14, 2014. arXiv: 1409.3215[cs]. URL: http://arxiv.org/abs/1409.3215 (visited on 08/23/2023).

[33] Unknown. *How does glibc malloc work?* 2023. URL: https://reverseengineering.stackexchange.com/questions/15033/how-does-glibc-malloc-work/15038#15038.

[34] Ashish Vaswani et al. „Attention Is All You Need“. In: *Advances in Neural Information Processing Systems* 30 (2017), pp. 5998–6008. (Visited on 08/23/2023).

[35] Michel Verleysen and Damien François. „The Curse of Dimensionality in Data Mining and Time Series Prediction“. In: *Computational Intelligence and Bioinspired Systems*. Ed. by Joan Cabestany, Alberto Prieto, and Francisco Sandoval. Red. by David Hutchison et al. Vol. 3512. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 758–770. ISBN: 978-3-540-26208-4 978-3-540-32106-4. DOI: 10.1007/11494669_93. URL: http://link.springer.com/10.1007/11494669_93 (visited on 08/30/2023).

[36] Donald J Wheeler. „Problems with Skewness and Kurtosis“. In: *Quality Digest Daily* (Aug. 1, 2011).

[37] Qiang Wu and Ding-Xuan Zhou. „Analysis of Support Vector Machine Classification“. In: *Journal of Computational Analysis & Applications* 8.2 (2006).

Additional bibliography

[38] Walter T. Ambrosius, ed. *Topics in biostatistics*. Methods in molecular biology 404. OCLC: ocn159977868. Totowa, N.J: Humana Press, 2007. 528 pp. ISBN: 978-1-58829-531-6.

[39] *CERT/CC Vulnerability Note VU#13877*. URL: https://www.kb.cert.org (visited on 08/30/2023).

[40] Vic Degraeve et al. „R-GCN: the R could stand for random“. In: *arXiv:2203.02424 preprint* (2022). URL: https://arxiv.org/pdf/2203.02424.pdf.

[41] Christofer Fellicious et al. *Machine Learning Assisted SSH Keys Extraction From The Heap Dump*. Version 0.1. Aug. 15, 2022. DOI: 10.5281/ZENODO.6537904. URL: https://zenodo.org/record/6537904 (visited on 09/06/2023).

[42] Vivek Gite. *How To Reuse SSH Connection To Speed Up Remote Login Process Using Multiplexing*. nixCraft. Aug. 20, 2008. URL: https://www.cyberciti.biz/faq/linux-unix-reuse-openssh-connection/ (visited on 10/21/2022).

[43] Jose Manuel Gomez-Perez, Ronald Denaux, and Andres Garcia-Silva. „Understanding Word Embeddings and Language Models". In: *A Practical Guide to Hybrid Natural Language Processing: Combining Neural Models and Knowledge Graphs for NLP*. Ed. by Jose Manuel Gomez-Perez, Ronald Denaux, and Andres Garcia-Silva. Cham: Springer International Publishing, 2020, pp. 17–31. ISBN: 978-3-030-44830-1. DOI: 10.1007/978-3-030-44830-1_3. URL: https://doi.org/10.1007/978-3-030-44830-1_3 (visited on 09/08/2023).

[44] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. URL: https://books.google.de/books?hl=en&lr=&id=omivDQAAQBAJ&oi=fnd&pg=PR5&dq=deep+learning&ots=MNV2eosBRS&sig=jN2QwFikq3g_YqU3hJVPEP0XIJ4&redir_esc=y#v=onepage&q=deep%20learning&f=false.

[45] Aidan Hogan et al. „Knowledge Graphs (Extended)". In: *ACM Computing Surveys* 54.4 (May 31, 2022), pp. 1–37. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3447772. arXiv: 2003.02320[cs]. URL: http://arxiv.org/abs/2003.02320 (visited on 09/08/2023).

[46] Weijie Huang and Jun Wang. *Character-level Convolutional Network for Text Classification Applied to Chinese Corpus*. Nov. 15, 2016. arXiv: 1611.04358[cs]. URL: http://arxiv.org/abs/1611.04358 (visited on 08/17/2023).

[47] Michael I Jordan and Tom M Mitchell. „Machine learning: Trends, perspectives, and prospects". In: *Science* 349.6245 (2015). Publisher: American Association for the Advancement of Science, pp. 255–260. URL: https://www.science.org/doi/full/10.1126/science.aaa8415.

[48] Ye Liu et al. „Kg-bart: Knowledge graph-augmented bart for generative commonsense reasoning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. Issue: 7. 2021, pp. 6418–6425. URL: file:///home/onyr/Downloads/16796-Article%20Text-20290-1-2-20210518.pdf.

[49] José Tomás Martínez Garre, Manuel Gil Pérez, and Antonio Ruiz-Martínez. „A novel Machine Learning-based approach for the detection of SSH botnet infection". In: *Future Generation Computer Systems* 115 (Feb. 1, 2021), pp. 387–396. ISSN: 0167-739X. DOI: 10.1016/j.future.2020.09.004. URL: https://www.sciencedirect.com/science/article/pii/S0167739X20303265 (visited on 08/30/2023).

[50] Dai Quoc Nguyen et al. „A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics, 2018. DOI: 10.18653/v1/n18-2053. URL: https://doi.org/10.18653%2Fv1%2Fn18-2053.

[51] Michael Schlichtkrull et al. „Modeling relational data with graph convolutional networks". In: *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*. Springer, 2018, pp. 593–607. URL: https://arxiv.org/pdf/1703.06103.pdf.

[52] Daixin Wang, Peng Cui, and Wenwu Zhu. „Structural Deep Network Embedding“. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco California USA: ACM, Aug. 13, 2016, pp. 1225–1234. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939753. URL: https://dl.acm.org/doi/10.1145/2939672.2939753 (visited on 09/11/2023).

[53] Liang Yao, Chengsheng Mao, and Yuan Luo. „KG-BERT: BERT for knowledge graph completion“. In: *arXiv preprint arXiv:1909.03193* (2019). URL: https://arxiv.org/pdf/1909.03193.pdf.

[54] W. Yurcik and Chao Liu. „A first step toward detecting SSH identity theft in HPC cluster environments: discriminating masqueraders based on command behavior“. In: *CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005.* CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005. Vol. 1. May 2005, 111–120 Vol. 1. DOI: 10.1109/CCGRID.2005.1558542.

[55] Jianlong Zhou et al. „Evaluating the Quality of Machine Learning Explanations: A Survey on Methods and Metrics“. In: *Electronics* 10.5 (Mar. 4, 2021), p. 593. ISSN: 2079-9292. DOI: 10.3390/electronics10050593. URL: https://www.mdpi.com/2079-9292/10/5/593 (visited on 09/11/2023).

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich diese Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe und alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind, sowie, dass ich die Masterarbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Passau, November 3, 2023

_____

Lahoche, Clément Claude Martial